Imperial College London

Department of Computing

Achieving High Quality Facial Rendering On Mobile Devices Using Volumetric Neural Rendering

Author:
William Thomson

Supervisor:
Dr Abhijeet Ghosh

Second Marker:
Prof Andrew Davison

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing of Imperial College London

August 2022

Abstract

State of the art capture for facial rendering typically involves large, expensive apparatus and a highly controlled capture environment, making it inaccessible to many. However, the growth of the 3D visualisation market means that the demand has never been higher for producing photo-realistic avatars. Therefore, this study aims to experiment with using mobile devices to capture facial geometry in less controlled environments with more accessible equipment. To achieve this, this study makes use of NeuS, a study on neural surface rendering, based on the highly popular neural radiance fields (NeRF) work. Our study finds that many challenges still exist for these *'in the wild'* capture pipelines, particularly lighting, which is still very influential on successful surface construction. However, this study demonstrates that results achieved through our NeuS pipeline, in these environments, produce significantly better surface meshes than those produced by existing structure from motion approaches with the same inputs. Additionally, through leveraging depth sensing hardware on mobile devices, we are able to reduce training times of NeuS by up to 4 hours, opening the door to exciting further research that can be done with depth data in this pipeline.

Acknowledgements

I would like to thank my supervisor **Dr Abhijeet Ghosh** for his guidance and ideas throughout this process. Additionally I would like to thank **Arvin Lin** for his time, support, encouragement and patience through our regular meetings.

Additionally I would like to thank my friends and family for their support, not only though this project but throughout my masters degree.

Contents

Chapter 1

Introduction

Computer representations of human facial avatars have been present from the early days of computer graphics in the 1960's and 1970's [1, 2]. The realism of these representations has increased year on year as hardware and software have improved. Over recent years commercial demand for producing photo-realistic 3D facial representations has seen a significant increase, driven by a growth in industries such visual effects, media and gaming. This demand is only expected to grow as increasing aspects of daily life are digitalised and investment in 'metaverse' technologies could see the 3D visualisation market grow at an annual rate of 12.7% over the next 5 years [3].

Despite the widespread digitalisation of the facial reconstruction process for many years, there is still often still a significant amount of manual work that needs to be done by 3D artists to produce high quality facial renders[4]. This increases both the time and the cost of the reconstruction process, and makes high quality capture inaccessible to many. Therefore, the desire to improve the facial reconstruction process and produce photo-realistic models programatically has captured the attention of the vision and graphics communities, both within the research and commercial worlds.

However, producing realistic facial models is difficult. Like many biological objects, faces have complex geometry, with uneven surfaces, mesoscopic details, non-uniform textures, as well as areas of occlusion from hair and orifices such as the mouth and nose [4]. Modelling this geometry requires not only high resolution rendering models for accurate representation, but also high fidelity data capture to reconstruct a subject without losing realism. Geometry is not the only challenge, facial skin also has complex reflective properties, and precisely representing how a face interacts with light is a crucial aspect of accurate reconstruction. Merely applying naive reflective models results in artificial-looking renders. In reality faces exhibit several reflective components, including specular, surface diffuse and subsurface diffuse reflection [5]. Hence, to create an accurate reconstruction, there are a lot of factors that need to be modelled. Finally, adding to this difficulty is the fact human faces are such a familiar sight, that even slight deviations from reality are perhaps more noticeable than in other objects [6].

Hence, there has been significant research in the fields of computer graphics and computer vision into the high quality capture and modelling of faces [7, 6, 8, 5]. The current state of the art apparatus for accurate facial capture is known as the 'light stage', a setup first proposed by Debevec et al in 2000 [7]. It requires cameras and lights to be fixed inside a hollow geodesic dome. A subject is positioned inside the dome, and photographed from multiple angles to build up a dense set of images, from which a reconstruction can then be created *(Figures 1.1 and 1.2, show light stages in use for the motion picture Spiderman 3)*. The advantage of these capture setups, is that controlled lighting can be used to illuminate the subject from a number of different directions. This makes it easier to inspect how the subject responds to different lighting conditions and enables the separation of specular and diffuse reflective components. Furthermore, the ability to exercise fine control over surface lighting enables the capture of surface geometry through various optical stereo methods, which facilitate 3D shape reconstruction.

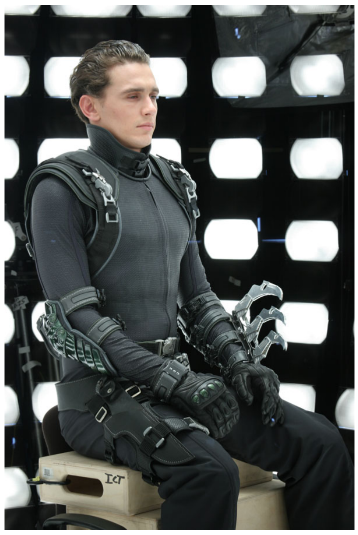


Figure 1.1: Actor James Franco being photographed in a light stage. [9]

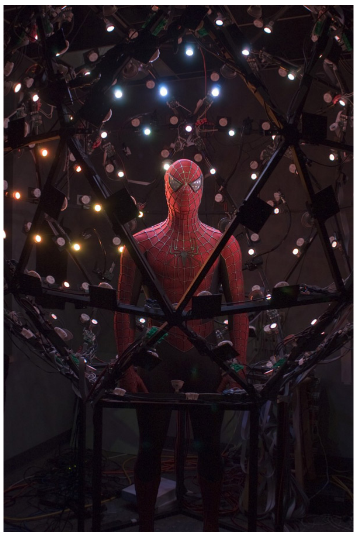


Figure 1.2: Actor being photographed in a light stage [9]

Whilst professional capture environments such as the light stage enable accurate facial reconstructions, they are also very inflexible and expensive, requiring a fixed setup and multiple expensive cameras and lights. Therefore, the goal of this project is to explore a more *'in the wild'* capture environment, one that uses handheld mobile devices to collect data for facial reconstruction. There has not been a significant amount of research into using mobile devices for facial rendering before, with the majority of studies into facial capture favouring high quality fixed DSLR cameras. The ability to produce renders from images captured on mobile devices would help to democratise the 3D reconstruction process, enabling those outside of research and professional capture industries to produce facial renders.

This study will primarily focus on **geometry** reconstruction, as opposed to reflectance estimation. To attempt this, this study will employ the use of neural volumetric rendering, combined with neural surface reconstruction. Neural volumetric rendering is a relatively new method for novel view synthesis and volume rendering, with the landmark paper Neural Radiance Fields (NeRF) released by Mildenhall et al in 2020 [10]. The method attempts to optimise a multi-layer perceptron to output the colour and density for a given 3D point based on its position (x, y, z) and viewing direction (θ, ϕ). Once the MLP has been trained on a set of input images it can then output colour and density for points in the 3D volume by querying the network with the (x, y, z, θ, ϕ) parameters, to render novel views. Whilst NeRF has achieved significant success, one area in which it can struggle is in the reconstruction of surfaces, particularly highly complex surfaces [11]. Hence, further studies have built on NeRF to include specialised surface reconstruction in its rendering pipeline [11, 12]. Neural Surface Reconstruction (NeuS) is one such method, and will be the primary method used in this study. NeuS builds on the core volumetric rendering method of NeRF, but it maps spatial position (x, y, z) to a signed distance function, and then uses the *zero-level set* of this network to represent the surface *S*.

1.1 Objectives

Overall, this project aims to experiment with using mobile devices, specifically, the iPad Pro, to capture data for facial geometry reconstruction using neural surface rendering. This will involve the following tasks:

1. **Develop a complete data capture and processing pipeline:**
   Build an iPad application for capturing video and depth data simultaneously. Then create an image processing pipeline for providing inputs to the NeuS rendering algorithm.

2. **Leverage depth sensing hardware to improve NeuS rendering pipeline:**
   Use the depth data captured by the mobile device to improve the NeuS rendering pipeline, whether that be through improving reconstruction quality or reducing training times.

3. **Evaluate NeuS render results with existing rendering methods:**
   Qualitatively evaluate the final renders generated from NeuS with renders produced by the Colmap structure from motion software.

1.2 Challenges

1. **Software Incompatibilities**
   One significant issue with this project has been issues with unsupported software. The most significant issue being Colmap, which was required to extract camera positions, as Colmap is not supported on the Apple Silicon M1 chips. This meant having to use the Department of Computing GPU machines, which did not have the dependencies installed for Colmap, hence this required liaising with the Department of Computing support staff in order to install the software with the required permissions.

2. **Long Training Times**
   The training time for NeuS, using the Department of Computing dedicated GPU cluster of NVIDIA Titan Xp GPUs, ranged between 14 and 25 hours. This meant that it was very difficult to iterate on new ideas or resolve bugs, as these issues sometimes were not apparent until training had completed.

1.3 Contributions

1. **Mask Generation:**
   Present and discuss a method for using depth maps to create masks for segmenting a subject's head and shoulders from the background.

2. **Interest Point Cleaning:**
   Present a method for automatic cleaning of Colmap-generated interest points to refine camera poses and area of interest bounds.

3. **Depth-informed ray sampling:**
   Using the depth data captured by the mobile devices to reduce NeuS training times by $\sim$3-5 hours, by providing a more refined ray sampling strategy.

1.4 Report Structure

First, in Chapter 2 there is a brief outline of some of the key prerequisite information required to understand the ideas discussed in this report. Chapter 3 then explores previous research in the fields of facial capture and volumetric neural rendering. Chapter 4 explains our capture and processing pipeline for producing the images, masks and depth maps to be fed to NeuS, as well as our novel, depth-informed, sampling method. Chapter 5 then discusses the observations made in this study and evaluates the results. Finally in Chapter 6 this report concludes its findings and suggests areas of potential further research.

1.5 Ethical Considerations

There are no significant ethical considerations that arise from this work directly. However, there is opportunity for digital human representation to be misused. The findings presented in this work make no attempt to aid acts of misuse, yet a clear and obvious consideration which must be made by anyone using or replicating this work, is to never capture representations without prior consent from a subject for the use of said representation.

Chapter 2

Background

2.1 3D Object Representation

There are several different ways to model objects in computer graphics. One division that can be made between the different methods is volume representations vs surface representations. As the name suggests, surface representations are used to display the outer surface of an object and act as an explicit boundary between inside and outside. Volume representations are used to represent the 3D volume of an object, which includes not only the surface but the space within an object.

The main benefit of surface representations over volume representations is that they are far less computationally expensive, enabling much faster rendering. Hence in the majority of computer graphics representations, particularly in cases where many objects need to be rendered, such as in video games, surface representations are favoured. Volume representations on the other hand are typically more computationally expensive, but contain more information about a scene and how the points in a scene relate to each other. Furthermore, one benefit of volume representations, the benefit utilised in this study, is that volume rendering is trivially differentiable. This enables neural optimisation, an idea which will be discussed in more depth when the neural volumetric rendering approaches themselves are discussed later on.

2.1.1 Volume Representations

Point Clouds

Point clouds are one example of volume representations. They are sets of data points with Cartesian *(x, y, z)* coordinates. Each point can also hold additional information such as colour or rotation information. This representation is typically less computationally expensive than other 3D representation methods. However, one of the issues with point clouds is that they do not also represent surfaces, instead a surface has to be extracted later on. For example by connecting nearby points to form a watertight mesh through algorithms such as Poisson surface reconstruction [13].

Voxels

Voxels are similar to point clouds in that they also discretise 3D space into a coordinate system, however these voxels are positioned regularly apart, unlike point clouds which can have non integer coordinates. These 3D voxel blocks can then be used to represent volumes. As with pixels, more complex shapes such as curves and fine details can be better represented by a higher resolution of voxels.

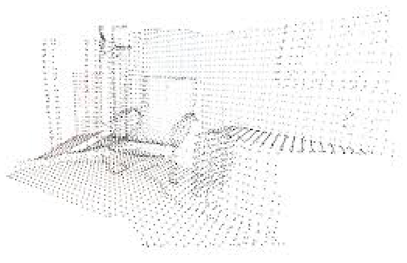


Figure 2.1: Point cloud [14]

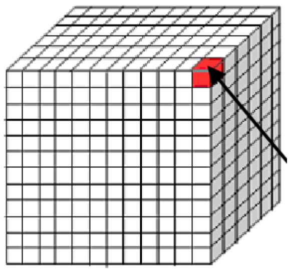


Figure 2.2: Voxels [15]

2.1.2 Surface Representations

Meshes

The most common method of surface representation in graphics are polygon meshes. These meshes are made up by a set of connected vertexes. On their own these vertices are similar to point clouds, however in a mesh, points are not represented individually but as a set of connected points, which make up an edge, and multiple edges can be used to represent a planar surface between points, thus constructing a watertight surface. Whilst mainly used to represent surfaces, meshes can also represent volumes via *volume meshes* which represent volume interiors as polygons.

2.1.3 Converting Volumes To Surfaces

Often volume representations can be used to represent a surface visually, for example if we only used a single layer of voxels we could represent a surface. However, in many cases, these volume representations have a lot of redundant data if the goal is to purely object representation. Therefore, it is useful to move from volume representations to surface representations, to improve computational efficiency when rendering. To make this conversion it is useful to think about representing a surface *implicitly* rather than *explicitly*. Implicitly being when we consider a surface as a function, $F(x, y, z) = 0$ or as a decision boundary, rather than an explicitly defined set of points.

Signed Distance Functions

One type of implicit representation is the signed distance function (SDF), these are functions that map points to the object's closest surface. They take a 3D position as input, and return as output, the distance from that position to its closet surface. Positive values represent points outside the volume, whilst negative values represent points within the volume.

Occupancy Networks

Occupancy networks are similar to signed distance functions, in that they consist of functions that take in 3D points, but unlike SDFs, these functions output the probability of a point being within the volume. Outputs which fall above the decision boundary are considered within the volume, and outputs which fall below the decision boundary are considered outside. This decision boundary can be considered to be the surface of the object.

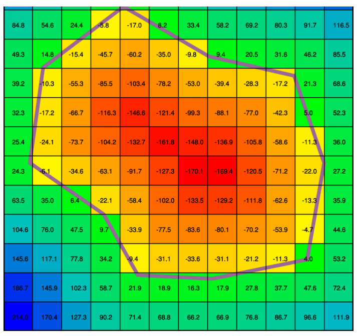


Figure 2.3: SDF [16]

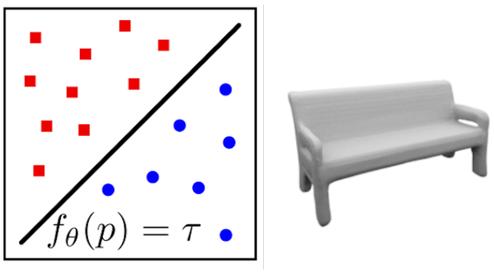


Figure 2.4: Occupancy network [17]

2.1.4 Level sets

Related to this idea of surface extraction are *level sets*. A level set is a set of values where the function has a constant value (for a real valued function). For a single variable x, this can be written as:

$$L_c(f) = \{(x_1, ...x_n)|f(x_1, ..., x_n) = c\}$$

A common example of a level set with two variables are the contour lines used on maps. When the level set has three variables, this can be used to represent a three dimensional surface. Hence, in the case of a signed distance function, the level set when $c = 0$ is the set of points which make up the object's surface.

2.2 Volumetric Ray Marching

Volumetric ray marching is a method for constructing 3D volumes which utilises the projection of rays through a 3D world space. One of the most well known methods which uses ray projection is *ray tracing* where rays are cast into a scene, and then the bouncing of the ray off different scene objects is simulated in order to estimate the radiance for each pixel.

A similar approach can be employed for volume rendering, except in this case, rather than calculate the rays bouncing off scene objects, rays are marched through the scene, passing through objects. The steps are as follows:

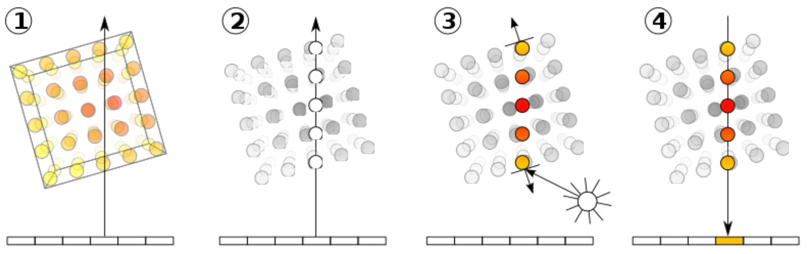


Figure 2.5: Steps of volume rendering [18]

1. Ray is fired through a 3D world space from origin point o along unit vector direction d.
2. Points are then sampled along the ray between the volume's *near* and *far* bounding space $r(p) = o + p * d$ whilst $p_{near} \geq p \leq p_{far}$.
3. Colour and density are estimated for each point/voxel the ray hits, in the case where a ray does not align with a point, an estimate is given, informed by the surrounding points *(e.g. by K-NN or trilinear interpolation)*.
4. These colours and density's are accumulated along the ray via process called α-compositing [19] to return the final ray colour .

The final ray colour can be computed as:

$$C(r) = \sum_{i=1}^{N} exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)(1 - exp(-\sigma_i \delta_i))c_i$$

Where N is the number of samples, σ and c are the density and colour estimations for sampled point i, and δ is the distance between two samples.

2.3 Multi-layer Perceptrons

Multi-layer perceptrons (MLPs) are the most classical type of artificial neural network (ANN). Artificial neural networks are a method of machine learning modelled around the biological neural networks found inside the human brain. These artificial networks are a collection of individual neurons (or perceptrons). At it's most basic level this neuron can receive a set of inputs x, with the weights w, then add a bias term b to output a value $\hat{y}$. Hence, the output can be written as:

$$\hat{y} = a(b + \sum_{i=1}^{N} x_i w_i)$$

Note that in this case the output is determined by the **activation function** a, which is function that can control what data is passed on by the neuron. A common example of this is the ReLU activation function $R(z) = max(0, z)$ (see Figure 2.7).

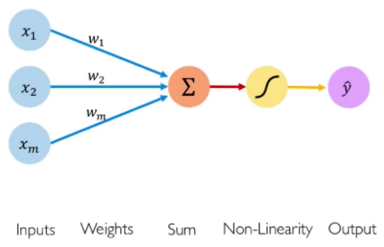


Figure 2.6: A single neuron with a set of inputs. [20]

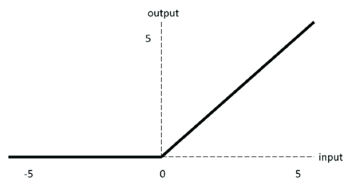
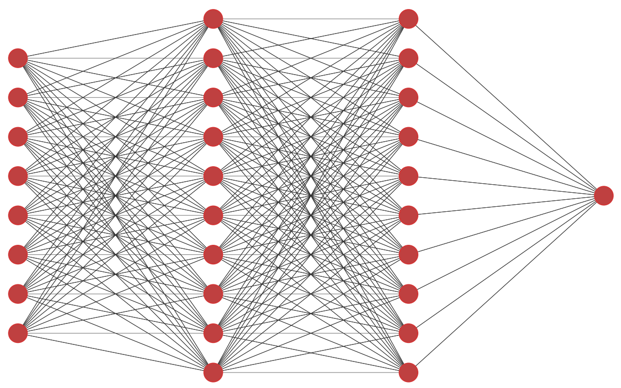


Figure 2.7: ReLU activation function. [21]

A single neuron can only represent a simple relationship, therefore, to model more complex functions multiple neurons can be connected in sequence. In these 'networks' of neurons the outputs of the previous layer act as the inputs to each neuron in the next layer (see Figure 2.8). These networks can be applied to model almost any relationship, hence ANNs are referred to as *universal function approximators*.



Input Layer Hidden Layer Hidden Layer Output Layer [1]

Figure 2.8: MLP with 8 inputs and 2 fully connected hidden layers

Backpropagation

When a network is first created, its weights and biases are all random. It is the updating of these weights and biases that constitute the training of the network, and enable the network to learn to produce a more accurate output. To achieve this, for each set of inputs, the network needs a *ground truth* value (y) against which its generated output ($\hat{y}$) can be compared via a *loss function*. A common loss function is mean squared error:

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

The gradient of this loss function with respect to the network weights is computed via chain rule to propagate back through network and update each of the weights.

2.4 Structure From Motion

Whilst not the core reconstruction method for this project, Structure From Motion *(SFM)* plays an important role in the neural volumetric rendering pipeline as a means of estimating camera poses. It is also used in the evaluation of our rendered outputs.

Structure from Motion is a method for estimating the 3D shape of an object from a set of 2D input images by estimating photometric consistency between images. The stages of the process are as follows:

1. **Feature Detection**: Detect interest points in each of the images, this might use common feature detection algorithms such as SIFT [22] or SURF [23] to produce unique feature descriptors for each point.

2. **Feature Matching**: Match interest points between pair of images. This involves using an algorithm such as K-nearest neighbours to match features which are most similar.

3. **Pose estimation**: Estimate the **fundamental matrix** which describes the correspondence between the two images. In Figure 2.9, this matrix describes the relationship between point x in camera C to epipolar line $l\prime$ in camera $C\prime$.

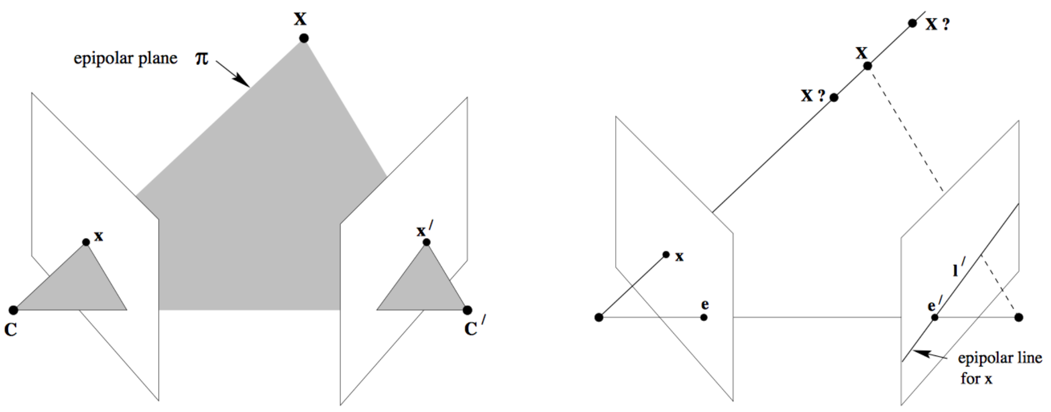


Figure 2.9: Epipolar geometry representation. Where x and $x\prime$ are the same point in two images, seen from cameras C and $C\prime$ respectively.

4. **Triangulation**: A 3D representation of the points in the image can then be built up via triangulation (see [24] for triangulation implementation details).

This process can be used to estimate other relative camera positions, given a first camera pose C.

Chapter 3

Literature Review

This chapter will contextualise the work carried out in this project by exploring the wider literature surrounding facial capture and neural volumetric rendering. In Section 3.1, the current state of the art for facial capture is discussed, exploring current methods and challenges. This section will look at the more traditional capture approaches in 3.1.1, which are typically more empirical, before exploring developments in the field of neural facial rendering in 3.1.2. Next, Section 3.2 introduces the field of neural volumetric rendering. This will lead into an examination into the two key studies behind this project, NeRF in 3.2.1 and NeuS in 3.2.2.

3.1 Facial Capture

3.1.1 Traditional Approaches

Facial Geometry

Modelling facial geometry has been an active area of research in computer graphics and computer vision for several decades [1]. Early work on facial geometry capture, particularly in the 90s and early 2000's, utilised 3D laser scanners to build up a dense set of 3D points of the head and face [6, 26, 27]. However, these scanners are very expensive and typically require a longer capture time, so they have fallen out of favour as other approaches have been developed. Furthermore, even with this scanning technology, particularly for early scanners, manual correction was still required to represent more complex geometry such as the mouth, nose and ears.

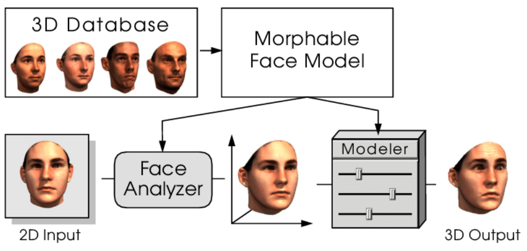


Figure 3.1: Morphable model creation pipeline from [25]. 2D input is used to inform combination of faces from 3D face database *(image from [25])*.

Moving forward from 3D scanners, an approach which has seen considerable use [28, 25, 29, 4, 30, 31] has been *3D morphable models*, which are datasets of scanned faces, each formatted as a shape vector. These faces can then be linearly combined to produce new face representations [32]. Morphable models have seen a resurgence in recent years with the growth of deep learning [4, 30]. These models have been able to produce particularly accurate results when using generative neural networks [33, 31], however typically have long training times.

One of the most commonly used techniques for facial geometry reconstruction, in current state of the art capture environments, has been *multi-view stereo* methods [34, 35, 36, 37, 38]. These techniques extract 3D scene information through identifying photogrammetric consistency between images and use this to extract depth and geometry. These techniques can be divided into *active* and *passive* methods, active methods being ones where some form of energy signal is projected onto a surface, and passive methods relying purely on appearance [39]. Passive, methods such as those used by Beeler et al [40] and Valgaerts et al [41] have been able to successfully capture facial geometry without controlled lighting conditions. However, these techniques sometimes struggle when modelling reflectance. Active methods on the other hand, such as photometric stereo, have proved effective for extracting high frequency geometry and reflectance [42, 36, 8]. These active methods have seen considerable use, particularly when using Debevec et al's light stage setup [7], due to the ease at which subjects can be viewed under different lighting conditions. However, work by Fyffe et al [42] demonstrates the ability to simulate these lighting conditions just using camera flashes rather than fixed lights.

The popularity of multi-view stereo methods has been largely due to their speed and accessibility, hence why they have been so prevalent in facial capture research and application. However, developments in new, neural, approaches are challenging the dominance of these methods.

Facial Reflectance

Whilst the focus of this report is primarily on reconstructing facial geometry, it is important to briefly mention the work on modelling facial reflectance, as this has been a major challenge in facial capture. Producing accurate facial models requires modelling the reflective properties of facial skin, a highly complex surface. Early attempts in facial reconstruction typically failed to accurately represent both specular and diffuse components, or applied view-dependant reflection models, which are unable to simulate accurate relighting [27]. Debevec et al's light stage provided a useful tool for measuring reflectance under many different views, and further studies have iterated on the initial light stage to produce more refined models [5, 8, 37]. This includes work by d'Eon et al [43] and Ghosh et al [5] to model subsurface scattering for increased relighting realism. Although, many of these studies use polariser filters to separate specular and diffuse components, hindering the portability of these techniques to mobile devices. However, exciting work by Kampouris et al. [36], extracts specular and diffuse components purely though binary gradient lighting, opening up the possibility for more portable capture equipment.

3.1.2 Neural Approaches

The explosion in the field of deep learning in recent years has influenced many aspects of facial capture. Hence, the 'neural' applications to the field encompass a wide range of research. As briefly mentioned previously, generative networks have been applied to the creation of 3D morphable models [33, 31], but deep learning has also seen significant application to areas such as 2D facial image synthesis [45] and facial animation [46]. These neural approaches have also been applied to traditional capture methods, for example work by Chandran et al [44] takes existing renders, captured through traditional stereo displacement maps, and then applies the StyleGan2 [47] architecture in order to improve the final output. The result was able to refine the rendered models particularly in areas such as hair and the inside of the mouth, which are areas empirical measurement approaches usually struggle with.

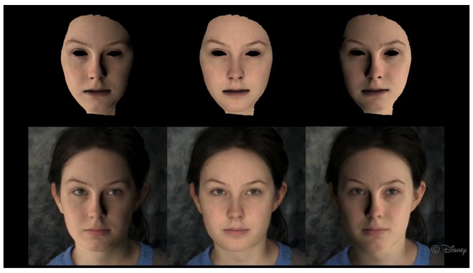


Figure 3.2: Work by Chandran et al [44] takes the ray traced renders shown in the top images and passes them through StyleGan2 to produce a more complete render *(image from [25])*.

The neural application which this project explores relates to **volumetric rendering** and **neural radiance fields** (NeRF) [10]. There has not been a significant application of NeRF-based methods for facial rendering, but some studies have explored the problem. Work by Gafni et al [48] combines Neural Radiance Fields with 3D morphable models in order to model both facial appearance and facial dynamics. This is done by utilising the implicit scene representation of NeRF to model facial appearance, with a low-dimensional morphable model to give control of facial expressions. Work by Sun et al [49] explores the use of using NeRF to produce 3D aware scene representation for 2D facial editing. They posit that volumetric methods produce better results for facial editing, due to the view-consistency offered by 3D aware methods, the same observation made by Gu et al [50]. However, these few works which have looked at using NeRF for facial reconstruction primarily focus on 2D novel view synthesis, rather than 3D surface reconstruction.

3.2 Neural Volumetric Rendering

Traditional rendering approaches, such as ray-tracing or rasterisation require detailed information about all of a scene's physical properties. This can include properties such as each object's geometry information, reflectance properties and global illumination conditions. These approaches produce highly realistic scene representations, however, they require all scene parameters to be known and provided as inputs. Neural rendering on the other hand, learns to estimate these scene parameters from already existing representations, such as images [51]. Hence these approaches can produce scene representations with far fewer inputs than traditional methods.

Neural volumetric rendering is a subsection of the wider field of neural rendering, and operates by adapting traditional volumetric rendering methods. Much of the process is the same, whereby rays are marched through a 3D world space, points sampled, then colour and densities composited. However, instead of the colour and density of the sampled point/voxel being known it is estimated using a multi-layer perceptron.

3.2.1 NeRF

Neural Radiance Fields (NeRF) is an influential piece of research by Mildenhall et al [10], for generating novel view synthesis from a sparse set of input images using a multi-layer perceptron to optimise a volumetric scene function. NeRF represents a 3D volume as a continuous 5D function $f(x, y, z, \theta, \phi)$, where (x, y, z) represent a 3D position and (θ, ϕ) represent the viewing direction. By optimising a feed-forward MLP to these 5 input parameters, NeRF predicts the density σ at each point, as well as the RGB radiance c of the point projected at viewing direction (θ, ϕ). Therefore, each image needs to have a corresponding camera pose, to represent viewing direction (θ, ϕ). Hence, in the case where camera poses are not known, camera pose estimation through techniques such as structure from motion are first required.

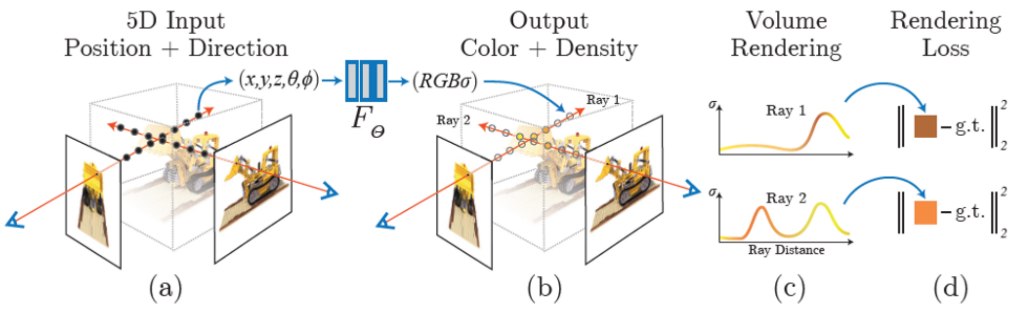


Figure 3.3: NeRF stages (from [10]). (a) sampling points along a ray and querying MLP, (b) getting the colour and density output for each point, (c) compositing colour and density for volume rendering, (d) calculating rendering loss from ground truth colour.

First, rays are marched through the scene and points along the ray are sampled. However, as the radiance and density of the points are not already known, they are instead estimated. To achieve this, the point (x, y, z) is passed into a fully connected multi-layer perceptron with 8 layers, which outputs σ (point density) and a 256-dimensional feature vector. This feature vector is then concatenated with viewing direction (θ, ϕ) and passed into one more fully connected layer, which then outputs c, which represents the RGB colour (see figure 3.4). The reason the viewing direction is passed into an additional layer to output colour is because NeRF makes the assumption that whilst density is only related to the 3D coordinate position, colour can be dependent on both position and viewing direction. The MLP loss function is then computed by calculating the total squared error between the rendered pixel colour in the generated view, with the true pixel colour, in the ground truth view from the same camera position.

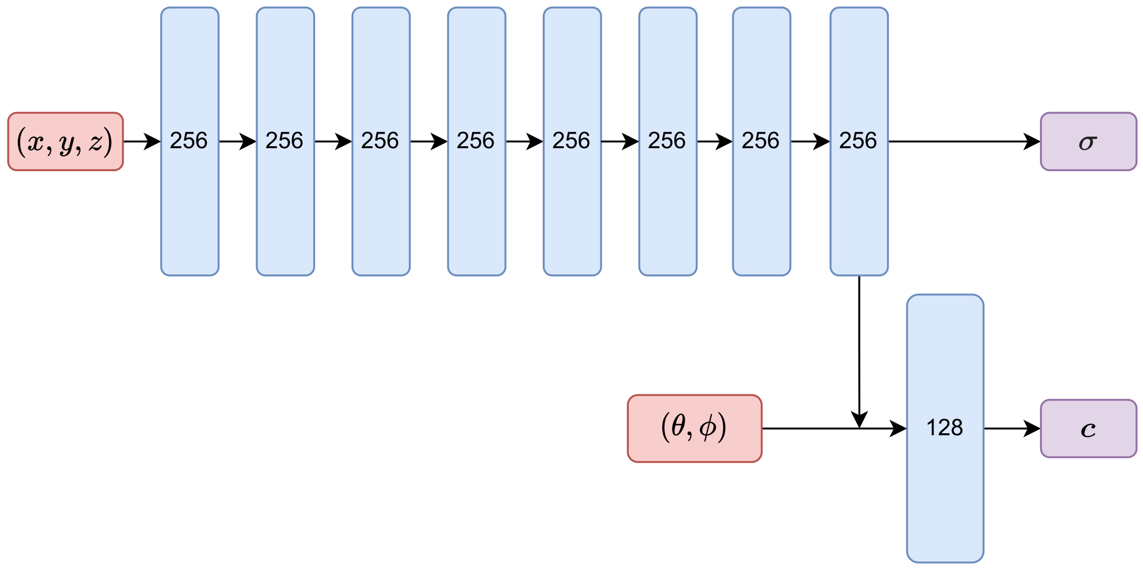


Figure 3.4: NeRF model architecture. Position (x, y, z) passed into 8 fully connected layers, each with 256 channels. Output of final layer is σ (density) and a 256-dimensional feature vector, which is passed into one more 128 channel layer alongside viewing direction (θ, ϕ) to output c (colour).

NeRF has spawned numerous further studies, improving and adapting the initial method. These include studies such as Mip-NeRF, which attempts to reduce blur by rendering canonical frustums rather than rays [52]. Another adaptation is KiloNeRF [53], which reduces NeRF's novel-view rendering times by querying multiple MLPs rather than a single MLP. One interesting novel area of research is exploring how NeRF can be adapted with depth estimation. Several studies [54, 55] explore the idea of estimating depth data from point clouds, produced via structure from motion, and using these to inform ray sampling in order to reduce the number of input images required for NeRF to represent a scene. Similar work by Dadon et al [56] discusses training a depth network to estimate scene depth maps in order to reduce NeRF training times. However, few of these studies have had access to ground truth depth sensor data.

3.2.2 NeuS

Whilst NeRF is capable of producing highly accurate 3D volumes, it can struggle when rendering complex surfaces [11]. Therefore, several studies have attempted to build on the core NeRF method, but with a focus on surface reconstruction [11, 57]. One such study is Neural Surface Reconstruction (NeuS), by Wang et al [57]. Wang et al note that state of the art surface reconstruction methods – in particular the Implicit Differentiable Renderer (IDR) approach developed by Yariv et al [58] – whilst good at representing complex surfaces, often struggle to model sudden changes in depth. On the other hand, volumetric rendering through NeRF is good at handling depth changes. Hence, NeuS attempts to combine the two approaches to achieve high quality surface reconstruction which can handle abrupt depth changes.

The NeuS reconstruction method is made up of two functions:

$$f : R^3 \to R$$

$$c : R^3 \times S^2 \to R^3$$

The first function f maps spatial position $x \in R^3$ to the point's signed distance from the object. The second function c maps the position of point $x \in R^3$ and view $v \in S^2$ to a colour. Both of these functions are encoded my MLPs. The main focus of this report is on surface reconstruction, hence we are primarily concerned with the function f. The surface S can then be extracted from this signed distance function by taking the zero-level set of this function $S = \{x \in R^3 | f(x) = 0\}$. The main difference between NeRF and NeuS, is that whilst NeRF encodes an MLP to output the density for a given point, NeuS takes this output to be the signed distance function. One issue with this is that NeuS still needs to make use of volumetric rendering. Thus the signed distance is converted into a point density via $\phi((f(x))$ where $f(x)$ is $x \in R^3$, and $\phi(x)$ is the logistic density distribution:

$$\phi_s(x) = se^{-sx}/(1 + e^{-sx})^2$$

This means that NeuS is able to make use of differentiable volume rendering, but to train an SDF network.

Chapter 4

Method

This chapter will explain the methodology used for capturing and processing the images used for neural surface reconstruction, as well our adaptations made to the neural surface reconstruction algorithm. First, Section 4.1.1 explains the image capture method, then Section 4.1.2 describes the image processing strategy, including depth representation, sampling strategy and mask generation. Section 4.1.3 then discusses the camera pose extraction approach. Finally Section 4.2 explores the NeuS library, describing configuration and explaining our depth-informed sampling strategy in 4.2.1.

4.1 Capture And Processing Pipeline

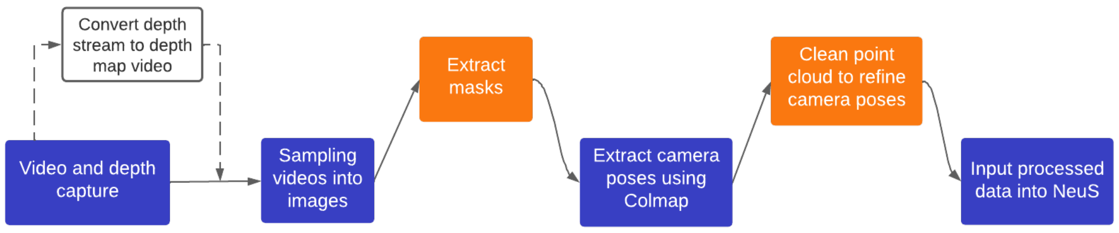

Convert depth stream to depth map video
Video and depth capture
Sampling videos into images
Extract masks
Extract camera poses using Colmap
Clean point cloud to refine camera poses
Input processed data into NeuS

Figure 4.1: Data capture and processing pipeline. Orange boxes represent novel contributions to the NeuS processing pipeline.

4.1.1 Data Capture

The first stage of the capture and processing pipeline involves capturing the raw image and depth data. This is achieved by recording a short video of the face using the front-facing camera of a 13 inch iPad Pro. The device's default camera application only captures RGB video, hence a custom application was built in *Swift*, utilising the Apple AVFoundation API to capture per-pixel depth data from the device's TrueDepth camera. This custom application simultaneously recorded RGB video at 30 frames per second with a 1800×2400 resolution.

The raw data outputted from the TrueDepth camera is a stream of 32-bit floating point per-pixel depths, for each frame, outputted at 30 frames per second. These depths are given as $\frac{1}{m}$ where m is the distance in metres away from the camera. This stream is read into a buffer on the iPad device, then compressed and written to the device's local storage. This depth stream can then be manually extracted. A potential further development here (from a software engineering perspective) is to automate this capture pipeline to upload the compressed stream directly to a web server for processing as opposed to manual extraction.

Capture Method

The goal of this project was to experiment with a capture environment which is uncontrolled, and attempts to explore the results which can be achieved using one person and a single device. For this reason, the image capture technique employed was to have a person hold the iPad and record their own face, even though having another individual record the face might have resulted in higher quality data. A recording was favoured over having a person take multiple photos, to try and minimise movement of the subject relative to the background between each image. The recorded video is later sampled down to a set of individual images.

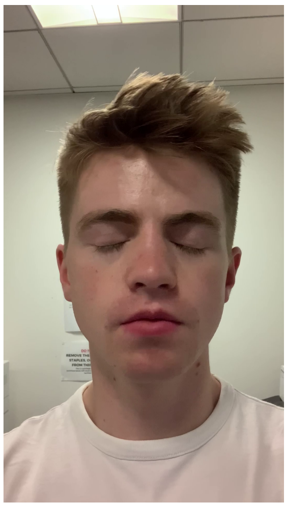


Figure 4.2

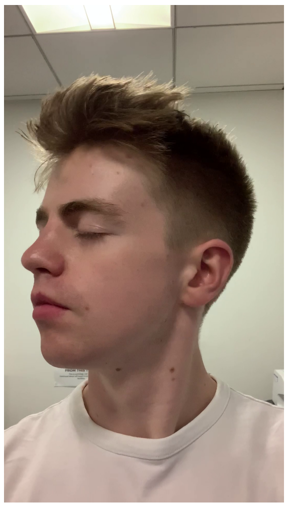


Figure 4.3

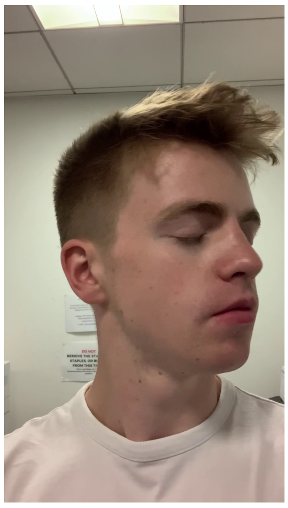


Figure 4.4

Two recording techniques were attempted, the first was to swivel/rotate the face, keeping the camera held still at the front of the face to capture different angles of the head (see Figures 4.2, 4.3, 4.4). Whilst this method is easy for the user it unfortunately led to issues when extracting the camera positions. The SFM software used to extract camera positions works by detecting and tracking interest points between frames. Even with a plain background the software can detect a lack of movement between images, most likely due to interest points detected on the neck and shoulders remaining fixed. The result was camera poses only being collected from the front of the subject, which is not suitable for reconstructing a complete face volume.

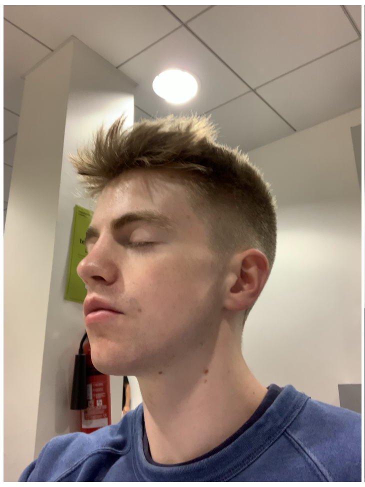


Figure 4.5

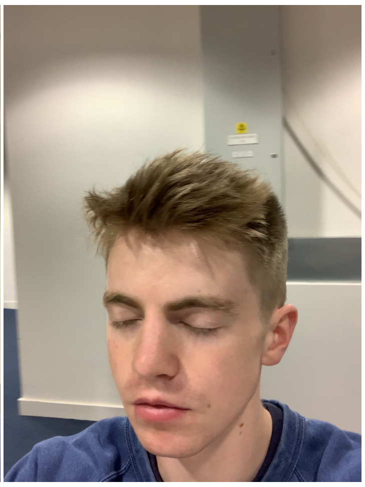


Figure 4.6

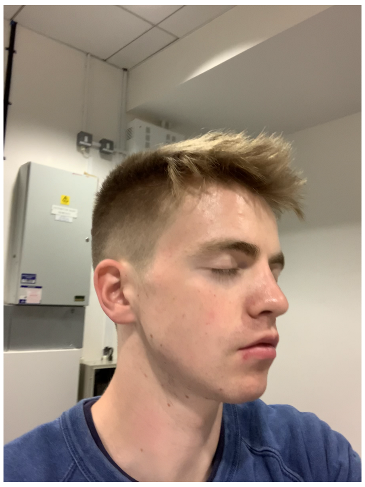


Figure 4.7

Therefore, instead of keeping the iPad still and moving the head, the method which was employed was to move the iPad around the face and make best efforts to keep the face and shoulders still (see Figures 4.5, 4.6, 4.7). One weakness of this method is that keeping the head and shoulders completely still whilst moving ones arms around to manoeuvre the iPad can be quite difficult, meaning there may be some discrepancy between images if one tilts the head whilst moving the iPad. However, this method was able to produce camera poses *surrounding* the target, which is more beneficial for using volumetric rendering to produce a complete face scan. A potential improvement here would be to use a smaller, more manageable device, such as a phone, as opposed to the large 13 inch iPad.

4.1.2 Image processing

Depth

Once the data has been captured, the next step of the pipeline is to present the depth and RGB data in a uniform format for sampling. Therefore the raw depth data is converted into an image format via the stages below:

1. The compressed depth data file is read in and decompressed.

2. The uncompressed raw depth data d is then normalised between 0-255 using $f(d) = \frac{d}{max(d)} * 255$.

3. A depth map video is created, whereby in each frame the pixel intensity represents the normalised depth data. Higher intensity pixels represent pixels closer to the camera (see Figures 4.8, 4.9 and 4.10).

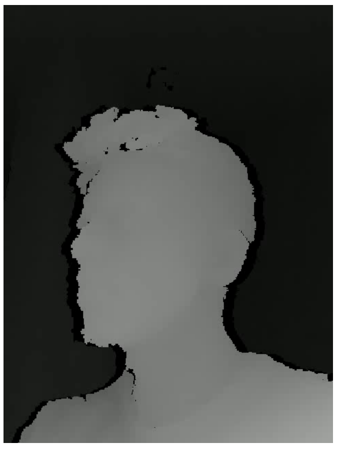


Figure 4.8

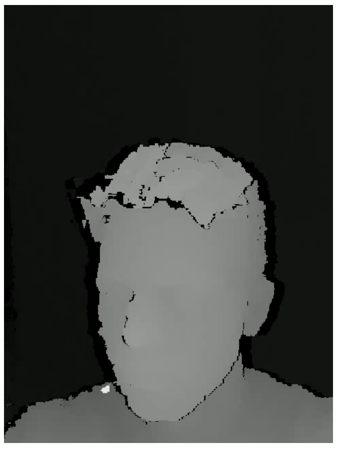


Figure 4.9

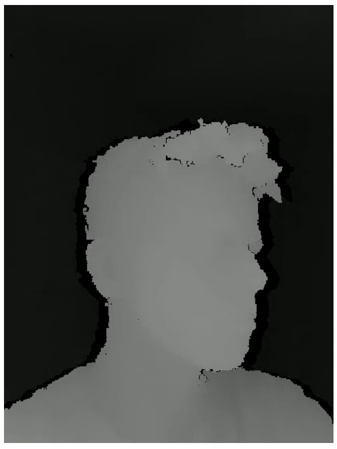


Figure 4.10

Sampling

Using all of the captured RGB images and depth maps would be redundant as there is very little movement between frames. Hence it would increase processing and training time but with little impact on reconstruction accuracy. Therefore, every 10^{th} video and depth frame was sampled simultaneously, resulting in $\sim$20-30 RGB images and $\sim$20-30 corresponding depth maps for a 10 second video.

Mask Generation

NeuS has the option to allow training with or without masks. Using masks aims to increase speed and accuracy by only sampling pixels inside of the mask. These masks are images, which have been encoded so that everything inside the mask has a white RGB value and everything outside the mask has a black RGB value (see examples in Figure 4.15 and Figure 4.17). This study explored using two different methods to create masks, the first was to use neural image segmentation, and the second was using an image's corresponding depth map to separate the head and shoulders from the background.

Neural Segmentation

The neural segmentation method we used utilised the open-source Facer library, (see Facer repository) which implements the findings of the RetinaFace paper [59], which uses a convolutional neural network and facial landmark detection to accurately segment faces in images. This model was applied to each of the input images to produce a corresponding mask leaving only the subject's head (see Figure 4.15).

Depth-based Segmentation

The second method this study explored was to utilise the depth images corresponding to each frame to create the masks. The mask generation process is explained below:

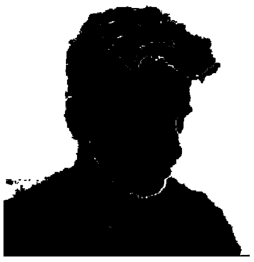


Figure 4.11

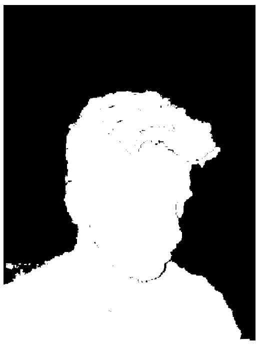


Figure 4.12

Figure 4.13

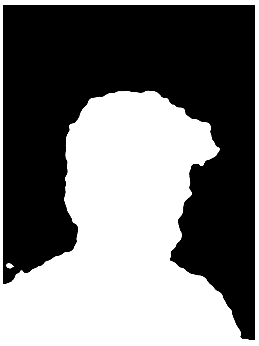


Figure 4.14

1. Calculate the threshold T by taking the mean pixel intensity across the image.
2. Encode the image based on T to colour values above T white, and values below T black, see Figure 4.11 *(note that this naive thresholding results in white holes in the mask around areas of sudden depth change such as the nose and hair)*.
3. To fix these gaps, first the thresholded mask is inverted, see Figure 4.12.
4. Floodfill the mask white from the top left corner, meaning every adjacent black pixel from this point is coloured white, leaving only the original mask holes coloured black, see Figure 4.13.
5. Then a bitwise $\&$ operation is applied to to combine the original thresholded mask in Figure 4.11 and the floodfilled image in Figure 4.13.
6. This produces a black mask with a white background, so a final inversion is applied, alongside a median blur with kernel size of 31 to smooth the mask edges, resulting in Figure 4.14.

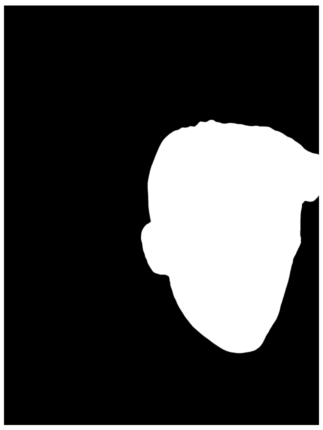


Figure 4.15

Figure 4.16: Example of a mask created using Facer segmentation.

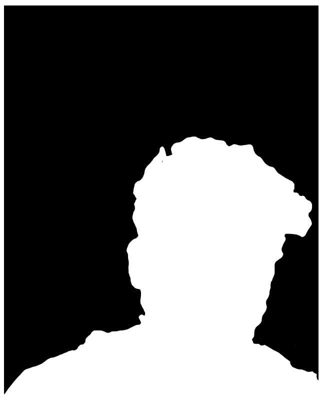


Figure 4.17

Figure 4.18: Example of a mask created from a depth map.

4.1.3 Colmap processing

Once the images have been processed, they are passed into the Colmap Structure From Motion software [60]. The Colmap software itself can be used to build up a dense point cloud representation of an object, which can then be converted into a surface representation via Poisson mesh construction. However, in this case Colmap is just used to acquire the relative camera poses for each image. These camera poses are currently unknown but are required for volumetric rendering in order to march rays through the 3D volume.

Functions for Colmap camera pose extraction are included in the NeuS library, as part of their data processing implementation. The NeuS functions first produce a sparse point cloud, then from this point cloud, extract relative camera poses. However, the built-in NeuS implementation produces a point cloud which often picks up background noise and struggles to isolate just the head. The issue with this is that the bounding volume for the rendering is too large (see Figure 4.19). In order to produce a more accurate construction, this point cloud needs to be refined to just encapsulate the region of interest (see Figure 4.20).

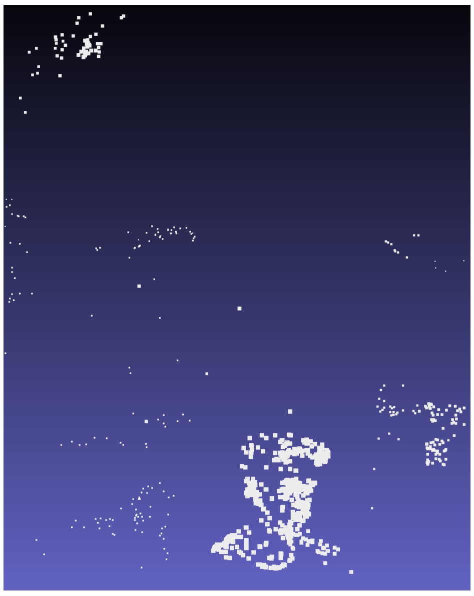


Figure 4.19: Noisy interest point cloud.

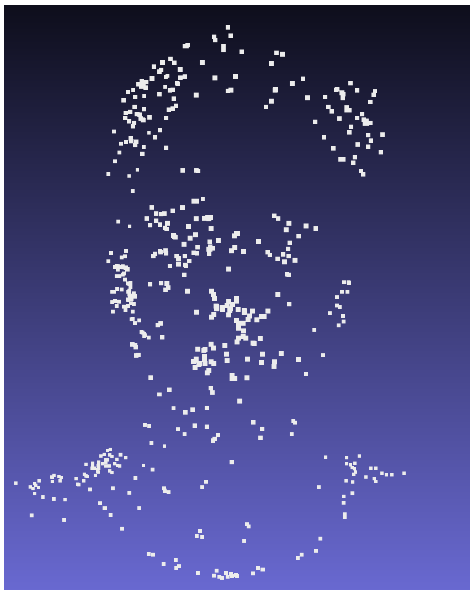


Figure 4.20: Refined interest point cloud.

Point Cloud Cleaning

The NeuS researchers, Wang et al [12], suggest manually cleaning this point cloud in software such as Meshlab. However, to improve this stage we proposed our own automatic cleaning method.

1. First calculate a central point $p = \frac{1}{n}\sum_{i=i}^{n} x_i$ for points x.
2. Next calibrate the location p, with hyper-parameter ϵ via $p + \epsilon$. We found that $\epsilon = (0, 2.5, 1.5)$ worked for all of our test cases.

3. Then calculate a sphere around this point with radius r and removed any points not within this sphere from the point cloud. We found the value $r = 5$ was able to always fully encapsulate the head.

This cleaning method is effective for the type of data we are collecting due to the fact we film *around* the target's head, meaning the head is present in all images. This means we have a dense set of points at the head, with a few sparse background points. The result of this cleaning was a point cloud which better enclosed the head and face (see Figure 4.20). Therefore, in the next stage of NeuS' Colmap processing, which is the extraction of a camera pose matrix from this point cloud, the camera poses were better able to encompass the area of interest, meaning the final render result was more isolated on the head.

4.2 NeuS

The final stage after processing the images, masks, depth maps and camera poses is to then pass that data into NeuS, the open-source neural surface rendering software developed by Wang et al [12]. The software allows the option of training with or without mask supervision. We ran a series of tests, training with our different types of masks and training with no masks. There are a number of parameters and configuration options for NeuS, the key training parameters used in this study are shown in Table 4.1. For more information about the NeuS configuration options see their paper [12] or their repository (see NeuS repository).

Iterations	Learning Rate	Batch Size	Ray Samples
300,000	$5e^{-4}$	512	64

Table 4.1: Iterations is the number of training iterations for the MLPs. **Learning rate** is the MLP learning rate for both the SDF network and the colour network. This learning rate starts at 0 and is then warmed up to $5e^{-4}$ over the first 5000 iterations. **Batch size** is the number of rays projected per batch. **Ray samples** is the number of samples taken along the ray which is marched through the volume.

Training the network on a cluster of NVIDIA Titan Xp GPUs can take between 14 and 25 hours, depending on the configuration and presence of any optimisation. This training time increased to between 30 and 40 hours when training on a single NVIDIA GTX 1080 graphics card. However, it is possible to extract information during training, which makes the process easier to iterate on. This is because NeuS outputs a volume mesh and a surface normal from a random view every 5000 iterations. Additionally it outputs a view validation, whereby the network generates a view from a known camera angle alongside the ground truth view from this angle. A surface can be extracted from the latest mesh by taking the 0 level-set of the most up-to-date SDF network.

4.2.1 Depth-informed ray sampling

NeuS uses volumetric rendering, in order to estimate the SDF for each point in the volume. Therefore, the first stage in the training process is to fire rays into the 3D space, in order to sample points for SDF and colour estimation. The ray emitted from a pixel is calculated as:

$$\{\mathbf{p}(t) = \mathbf{o} + t\mathbf{v} | t \geq 0\}$$

Where **o** is the center of the camera from which the ray is being fired and **v** is the unit direction vector of the ray. During training NeuS generates a random set of 512 rays per camera pose and samples 64 points along each ray.

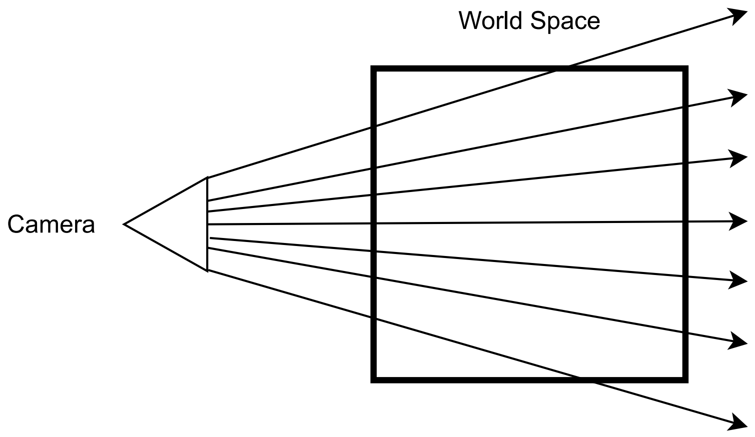


Figure 4.21: A representation of the projection of rays from a camera position into the world space.

NeuS ray sampling assumes the region of interest for each ray to be within a unit sphere. To calculate this sphere first a midpoint m is calculated, then the near and far bounds of this sphere, n and f, are calculated as $n = m - 1$ and $f = n + 1$. The initial midpoint is calculated using the camera center (**o**) and ray direction (**v**) via:

$$m = \frac{o \cdot v}{v \cdot v}$$

The result of this implementation is a concave set of midpoints $\{m\}$, each of which is bound by a sphere with a near and far bound, n and f. When sampling, 64 points are taken at a uniform distance along this ray. Figure 4.22 shows how the midpoint of this bounding sphere moves through the volume as different rays are sampled. Figure 4.23 shows how $\{m\}$ form a concave plane from the camera viewing direction.

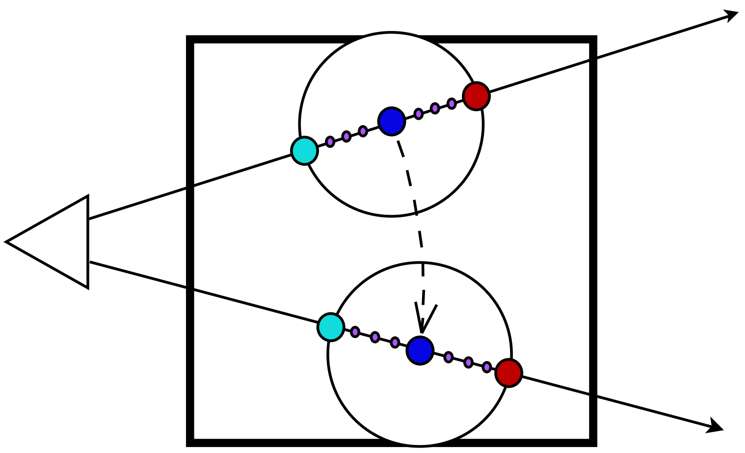


Figure 4.22: Representation of how NeuS forms bounding spheres around the calculated midpoints and then samples points within those bounds. The dotted line represents the movement of the midpoint through the scene as new midpoints are calculated for each ray.

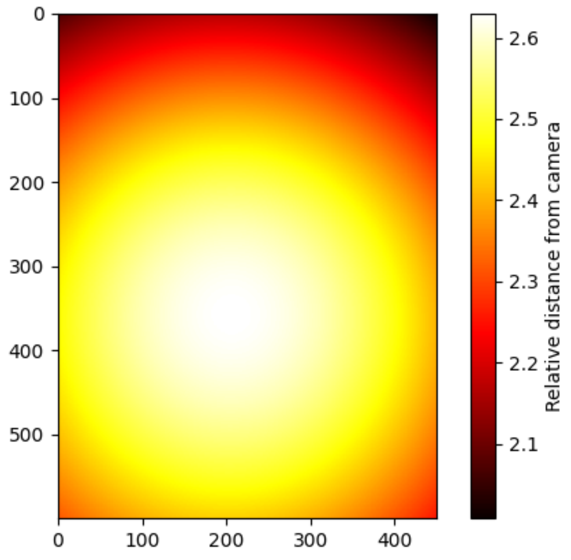


Figure 4.23: Heatmap showing the distance of midpoint m away from the viewing camera in the original NeuS implementation. Note the image here has been converted to a (450, 600) resolution from (1800, 2400).

The naive bounding strategy employed by NeuS is able to accurately capture the area of interest. However, this implementation makes no assumption about the geometry of the target object. Therefore, to attempt to improve on the original NeuS approach, we formulate a method for a more geometry-based ray sampling. The goal of this approach is not to calculate midpoints arbitrarily, like NeuS does, but to base our midpoint calculations off the distance to the object's surface, see Figures 4.24 and 4.25 for a representation of this.

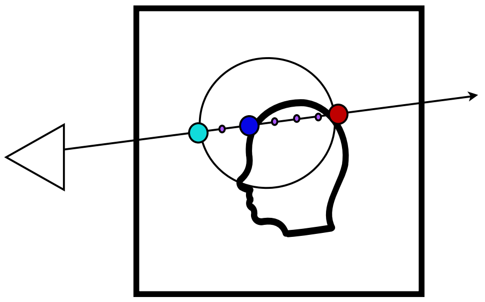


Figure 4.24: Depiction of geometry-informed midpoint, near, and far bound estimation.

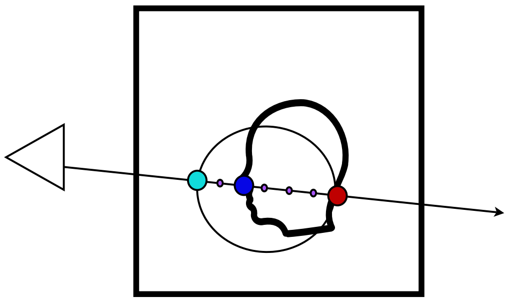


Figure 4.25: Depiction of geometry-informed midpoint, near, and far bound estimation.

In order to implement a more geometry-informed approach, the per-pixel depth data is utilised to estimate a midpoint m' which is based on the object's distance from the camera. To achieve this we first invert the depth data – so that larger values represent pixels further away – and then normalise it. This processed depth d' can be calculated as $d' = \frac{(255-d)}{255}$. Figure 4.26 shows this d' for a random input view. To combine d' with the naive midpoint estimation m, we multiply d' by m, to give our adapted midpoint estimations m', these are represented by Figure 4.27. The reason we multiply the two rather than just use raw depth, is so that we can output an estimate which is more aligned to the 3D volume's coordinate range.

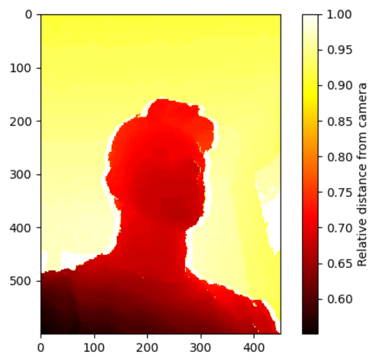


Figure 4.26: The depth data which corresponds to an input image after being inverted and normalised between 0-1.

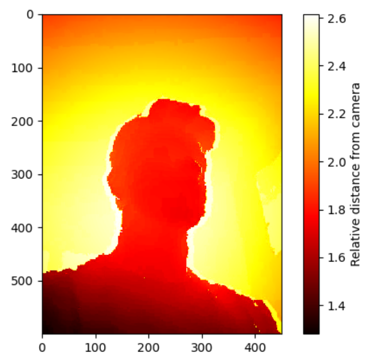


Figure 4.27: The combination of the original NeuS midpoint estimation with the depth data.

The aim is to have these new midpoint estimations give more refined sampling points. Therefore, we reduce the number of samples per ray from 64 samples to 32 samples, and reduce the size of the near sampling bound to $n = m - 0.5$. This is because with a a geometry informed midpoint, we expect anything closer than that midpoint to be empty space (see Figures 4.24 and 4.25 for a depiction of this).

Chapter 5

Discussion and Evaluation

This chapter discusses the final results of our capture process, particularly how the decisions and alterations made in our capture pipeline impacted the final mesh result. The main objective of this study is to assess the surface geometry reconstruction. Therefore, whilst there is some discussion of 2D novel view synthesis, the majority of the discussion will examine texture-less surface meshes.

Section 5.1.1 explores the results achieved when using different mask generation techniques. Section 5.1.2 discusses the influence of different lighting conditions in the capture environment. Section 5.2 then explores the impact that our novel depth-informed ray sampling had on mesh results and training speeds. Finally, in Section 5.3 our surface mesh results are compared to surface meshes acquired through the Colmap structure from motion software.

Quantitative evaluation of the final mesh quality is difficult without a ground truth 3D mesh, hence the majority out of the evaluation in this chapter will look to qualitatively discuss mesh quality. However, the impact of the depth-informed ray sampling is quantitatively evaluated against the standard sampling method.

5.1 Pipeline Discussion

5.1.1 Influence of Masks

In our image processing pipeline we used three different mask approaches. These were *depth-map masks*, *neural image segmentation (Facer)* and *no masks*. The neural image segmentation technique used the open-source Facer library, based on work by Deng et al [59]. This resulted in masks only including the head and face. The depth-map masks were a contribution introduced by this report, to segment the head, neck and shoulders from the background using the corresponding depth map captured for each image. The type of mask used impacted a number of areas including, processing speed, mask generation stability and final mesh surface quality.

Processing Speeds

Generating any form of mask increases the time of the processing pipeline. The type of mask generation technique used had a significant impact on how much additional time was required. We found that the Facer masks take considerably longer to generate than the masks generated using the depth maps. This additional time is to be expected, as the Facer library must first query a convolutional neural network to find face locations, and then parse this area to segment the face and hair. Depth-map generated masks on the other hand only require pixel thresholding and colouring. Hence, the depth-map generated mask take an average of 39.2 seconds, whilst the Facer masks take an average of 712.8 seconds to run, when processing 20 images.

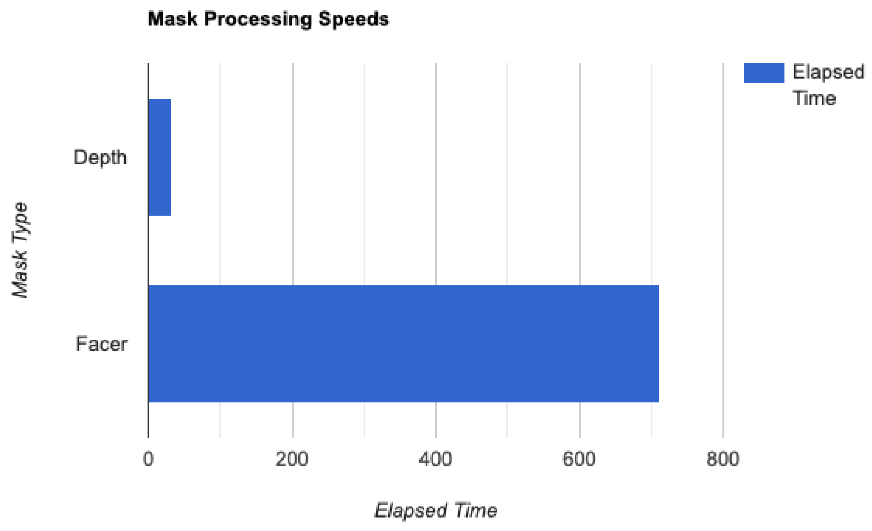


Figure 5.1: Processing times for depth map masks vs Facer masks, averaged over 10 attempts, each processing 20 images. Depth mask time is only 39.2 seconds, whilst Facer mask is 712.8 seconds.

Mesh Results

The type of mask used had a noticeable impact on the final mesh result. Examples of the extracted surface acquired with the same data, but using, no masks, depth-map masks and Facer masks, can be seen in Figure 5.2, Figure 5.3 and Figure 5.4 respectively. The worst quality mesh was the mesh created using the depth-map masks, with Facer mask and no mask configurations producing higher quality results, but each still with their flaws. It could be argued that the mask-trained models produced lower quality meshes, thus questioning the necessity of masks. However, one of the benefits of using masks is a faster training process, with the average training time when using masks being 18 hours, compared to 24 hours when training without masks.

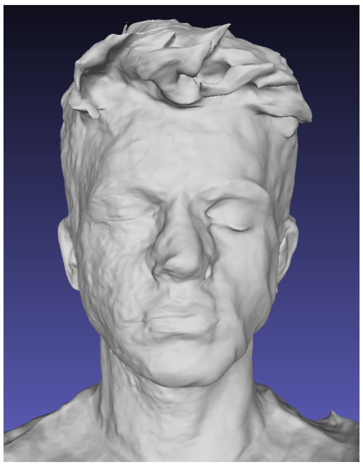


Figure 5.2: Mesh captured with no masks.

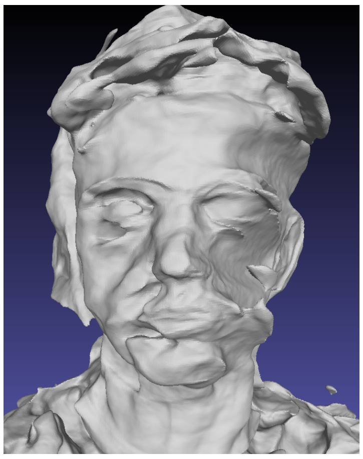


Figure 5.3: Mesh captured with depth masks.

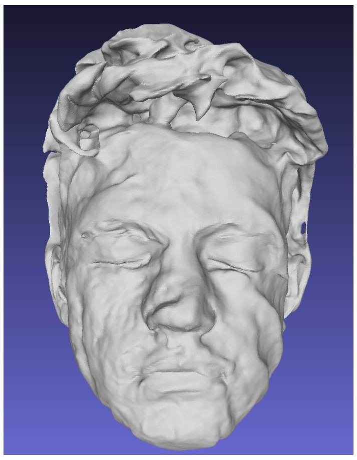


Figure 5.4: Mesh captured with Facer masks.

The two highest quality meshes produced were when training with no masks and when using the Facer masks, yet both meshes still contained multiple inaccuracies. The meshes produced using the Facer masks include a smaller area of interest and these meshes tended to accentuate fine details, such as the eyes, eyebrows and lips, see Figure 5.5. At times this accentuation of details could lead to increased artifacts and inaccuracies on the skin, for example, see the bump above the right eye in Figure 5.4 and the lines on the right cheek in Figure 5.7. On the other hand, the no-mask meshes typically captured a larger area, and had fewer large artifacts, but also had more blurred surface details. This can be seen on the right-hand cheek and eye in Figure 5.8. The differences in the two meshes is most likely due to the different size of the interest areas. Hence it is possible that the Facer mask volumes are being over-sampled in certain areas, whilst the depth mask volumes are being under-sampled in certain areas. Definitely determining a better mesh is difficult in this case, due to the qualitative nature of the evaluation, however, on balance, the no-mask meshes appear to produce the best results.

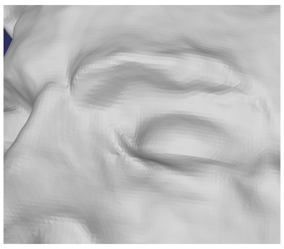


Figure 5.5: Facer mesh.

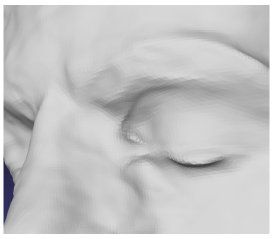


Figure 5.6: No-mask mesh.

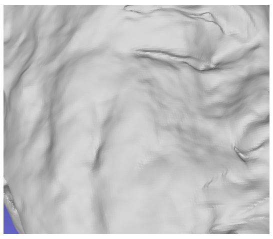


Figure 5.7: Facer mesh.

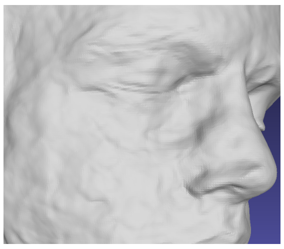


Figure 5.8: No-mask mesh.

The most noticeable difference in quality was between meshes created using depth-map masks and the other approaches, with depth-map masks typically resulting in a low quality of mesh with many artifacts, see Figure 5.3. A likely reason for this is the rough mask edges caused by the uneven depth data around the edges of a subject. This can be see in Figure 5.9 and Figure 5.11, around the person's shoulders and ears. This rough texture can result in discrepancies between views, meaning areas are sometimes included in the interest area, but sometimes not. Thus, artifacts are created where the MLP has not been able to correctly learn the SDF for a point, due to a lack of samples for that point. We attempted to combat these rough mask edges in our image processing by smoothing the edges of the mask with a median blur. However, as shown in Figures 5.10 and 5.12, the edges still suffer from a choppy texture even after blurring. Further smoothing of the mask can be dangerous because it allows more areas outside of our interest area to be included in the mask. This can lead to situations such as the one in Figure 5.17, where the mesh fuses with the background.

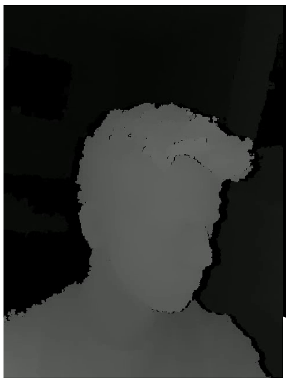


Figure 5.9

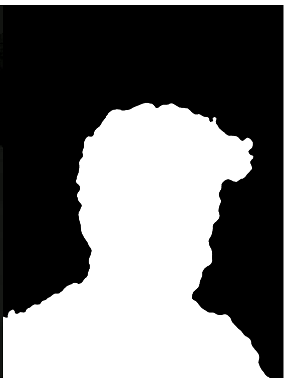


Figure 5.10

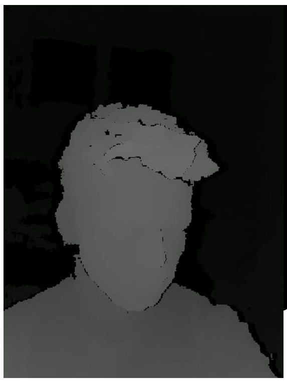


Figure 5.11

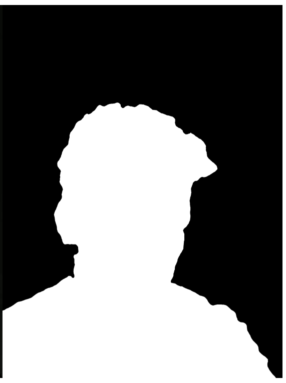


Figure 5.12

Overall, we found that a more influential factor on successful surface reconstruction, as opposed to using masks, was extracting camera poses which accurately define the area of interest, as was explained in Section 4.1.3 of our Methodology Chapter. We recommend that further work on this pipeline should prioritise accurately extracting camera poses over mask generation.

Stability
Despite the Facer masks producing higher quality geometry than the depth-map generated masks, the Facer masks were at times prone to errors in face segmentation. Although these errors were quite rare, occurring only 4 times when processing 300 images. Examples of these errors can be seen below, Figure 5.14 being an error segmenting Figure 5.13 and Figure 5.16 being an error segmenting Figure 5.15 .

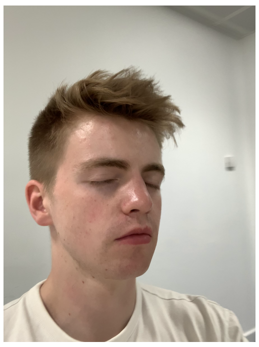


Figure 5.13

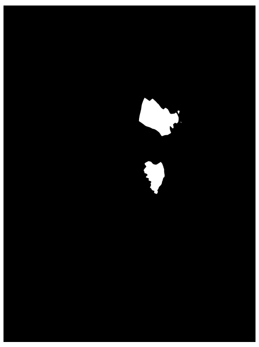


Figure 5.14

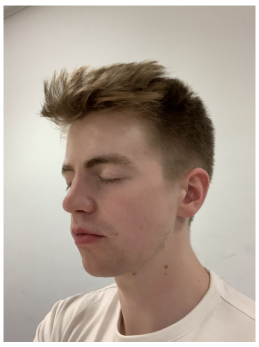


Figure 5.15

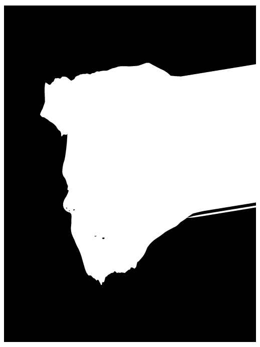


Figure 5.16

The result of these mask segmentation errors has a very significant effect on the final render result, particularly if these errors extend the mask beyond the head. This results in this additional area being incorporated into the final mesh, often fusing the background with the head mesh itself, see Figure 5.17. We found depth maps to be more reliable, as even when there is not a significant depth difference between a subject and their background, this is reflected in a higher average inverse depth, which is used to threshold the image. Hence, further work on improving the edge roughness of the depth-map masks could see them become a more stable, faster alternative to Facer masks.

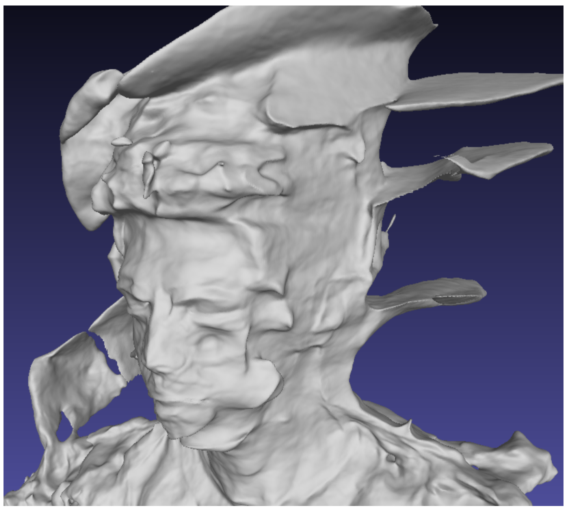


Figure 5.17: Example of mesh produced when input masks fail to properly segment face.

5.1.2 Influence of Lighting

Point-Light Sources

This study found that one of the biggest influences on mesh quality was the lighting conditions of the capture environment. One capture environment used in this study consisted of a space with several overhead point light sources. Examples of the images captured in this environment can be seen in Figure 5.18 and Figure 5.19. In this environment the closest light source was situated above the right-hand side of the head. This resulted in heavy shadows being cast across the left-hand side of the face, particularly around areas of sudden depth change such as the eye socket and outside of the nose. In fact, using masks we calculated that the average pixel intensity was 14 units higher on the right hand side of the face than on the left hand side of the face in this environment (see Figure 5.30). The reconstruction achieved using this set of images produced highly inaccurate mesh with many areas of poor reconstruction, where the SDF network has failed to accurately estimate depth. This mesh can be seen in Figure 5.20 and Figure 5.21.

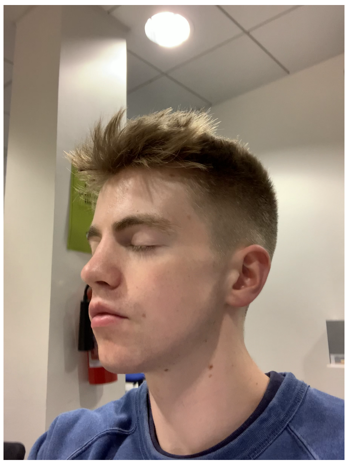


Figure 5.18

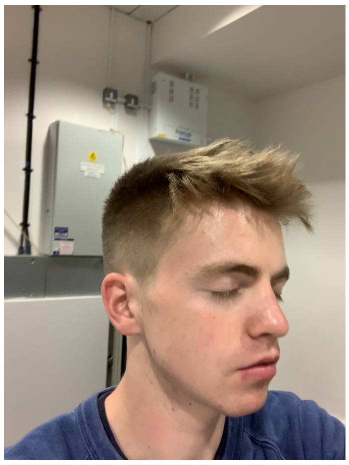


Figure 5.19

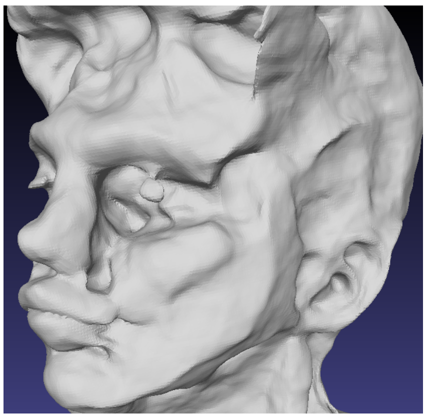


Figure 5.20

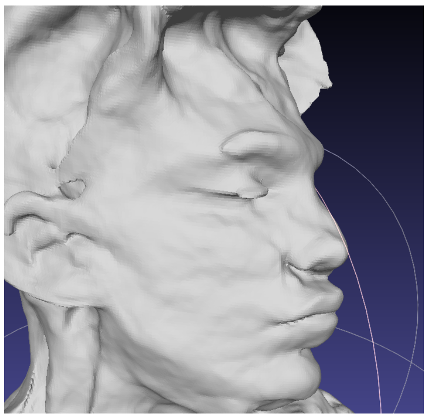


Figure 5.21

As shown in Figures 5.20 and 5.21. Areas of the mesh have significantly inaccurate depth, most noticeably the left-hand eye socket in Figure 5.20. The most likely reason for this is that NeuS' SDF estimation struggles to find a local minimum, due to the heavy shadow darkening all of the colours in the area. These shadows result in NeuS assuming a greater depth than actually exists in the area. However, whilst this lighting had a significant impact on surface reconstruction, it had less of an impact on novel view synthesis. Figure 5.22 shows a view created by NeuS, and Figure 5.23 shows the ground-truth view from the same camera position. As observable here, whilst the overall render is a little blurry, there is no significant error in the render, as exists on the mesh. Additionally, the right and left sides of the face are of similar render quality, unlike with the surface construction, which is of a significantly lower quality on the darker side of the face.

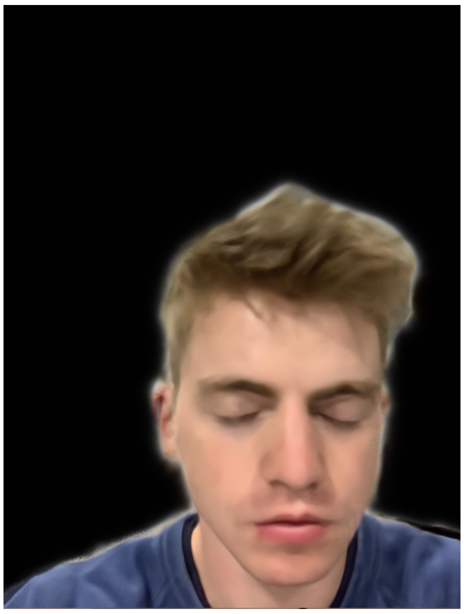


Figure 5.22: A 2D view rendered by NeuS using the fully trained MLP, from the same camera pose as Figure 5.23

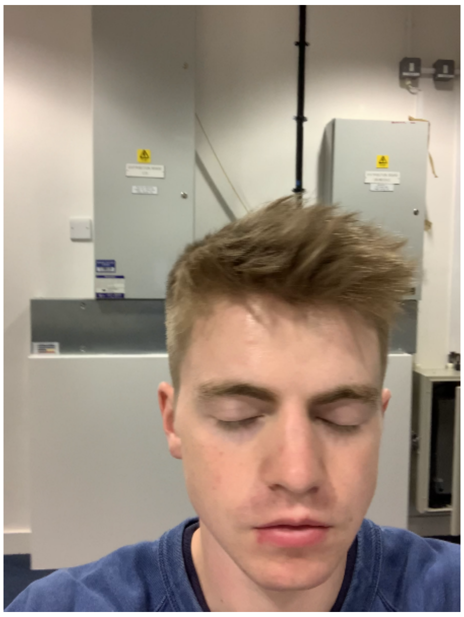


Figure 5.23: The ground truth view from our input images, used to validate Figure 5.22

Ambient Light Sources

After capturing images under point light sources, we decided to explore how more ambient lighting conditions would impact surface reconstruction. Figure 5.24 and Figure 5.25 show images acquired under more ambient lighting conditions where the front of the face is illuminated by natural light entering through a large window, but without direct sunlight to cast heavy shadows. As shown in these images, pixel intensity is more consistent across the face, with fewer heavy shadows and more uniform colouring. We used masks to calculate that the average pixel intensity on the right side of the face was only 3 units higher than on the left hand side (see Figure 5.30 for a comparison with images captured under point light sources).

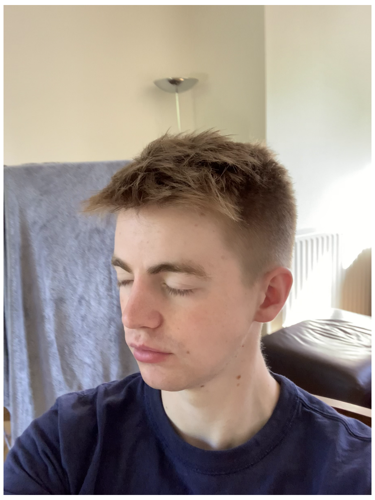


Figure 5.24

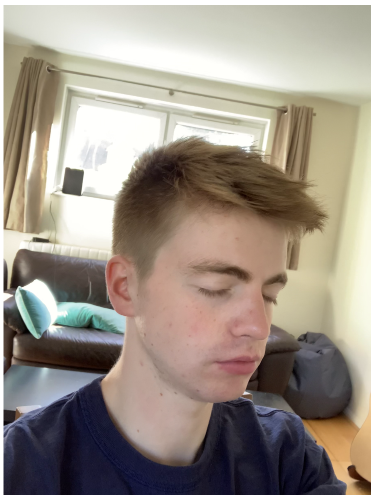


Figure 5.25

The result of capture under ambient lighting conditions is a much more uniform and accurate mesh, as shown in Figure 5.26 and Figure 5.27. This, mesh did not suffer from the sudden holes around the nose and eye sockets, as the mesh acquired under the point light sources did. Whilst there is not as much discrepancy between left and right sides of the face with this mesh, there are still some differences. The right side of the face is a little more blurry with a rougher texture, whilst the left side is smoother, yet has the occasional large artifact. Interestingly, this improved lighting condition seems to have little influence on 2D novel view synthesis. Figure 5.28 is a view synthesised by NeuS, from the same camera position as the ground truth image show in Figure 5.29. As this image shows, it is of similar quality to the novel view generated in the point lighting conditions, with a slight surface blur but no noticeable rendering errors.

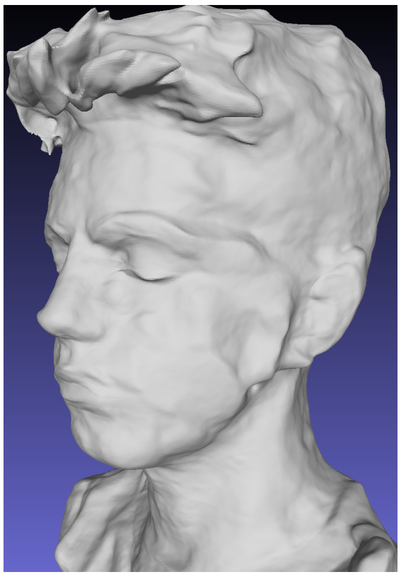


Figure 5.26

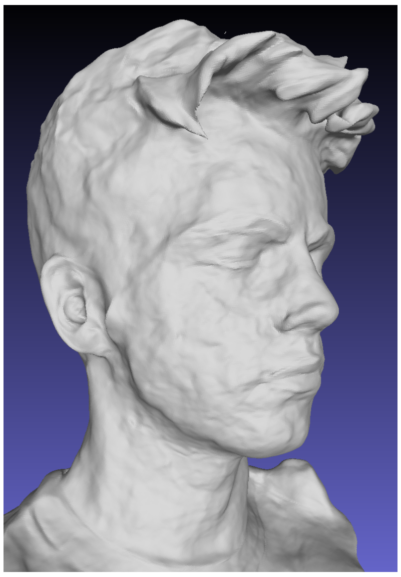


Figure 5.27

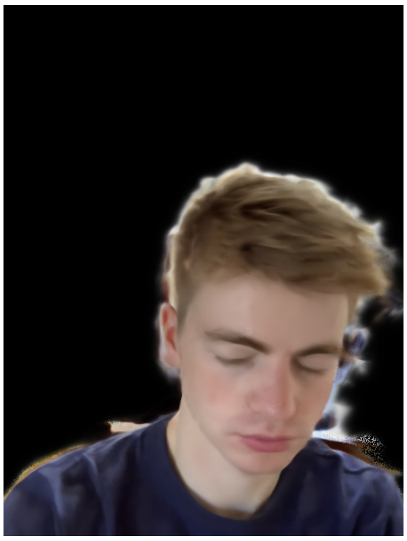


Figure 5.28: A 2D view rendered by NeuS using the fully trained MLP, from the same camera pose as Figure 5.29

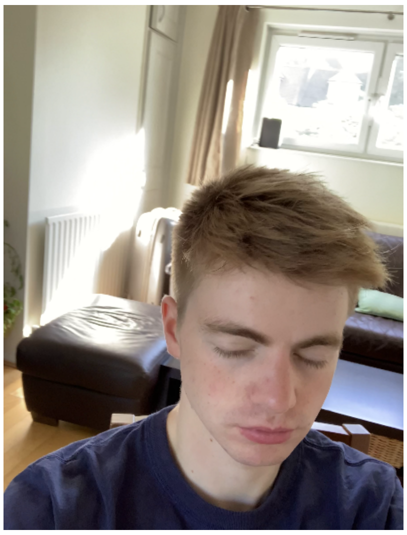


Figure 5.29: The ground truth view from our input images, used to validate Figure 5.28

The reliance on more ambient, uniform lighting conditions for accurate mesh construction is one of the biggest limitations of this capture pipeline, as our goal is to achieve high quality in less controlled environments. However, see the future work Section 6.2 for more on how future studies could attempt to overcome this issue.

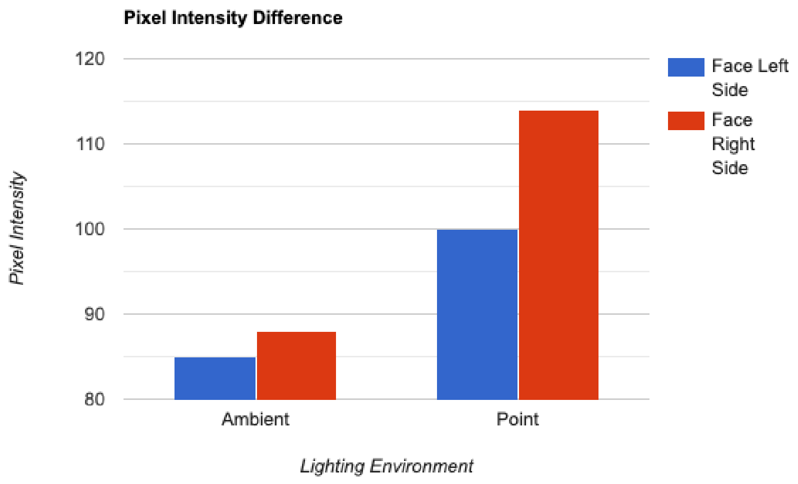


Figure 5.30: The average pixel intensity of the left and right face areas plotted against each other, for point and ambient lighting environments.

5.2 Depth-Informed Ray Sampling

This study attempted to improve the original NeuS ray sampling strategy by providing a more refined estimation for the near and far sampling bounds. This was achieved by using an image's corresponding depth map to inform the midpoint m of the interest area, and then reduce the distance of the near bound n, so to reduce samples on empty space. The goal was to use this informed sampling to reduce the number of sampling points and thus thus speed up the NeuS network training.

Configuration	**Time (hours)**
No-mask supervision	25
No-mask supervision *(depth-informed)*	21
Mask supervision	17
Mask supervision *(depth-informed)*	14

Table 5.1: Training times for different configurations when using depth-informed sampling and when using original NeuS sampling

Halving the number of samples from 64 per ray to 32 per ray had a significant impact on training times. Overall, training times were reduced by ~3-4 hours, depending on the configuration, with the quickest time now being 14 hours. However, it is difficult to definitively say how effective our depth sampling strategy was. The loss rate of the SDF network for the original NeuS sampling strategy and our sampling strategy is shown in Figure 5.31. As this chart shows, the depth-informed sampling had little impact on the loss convergence in comparison to the original NeuS sampling strategy.

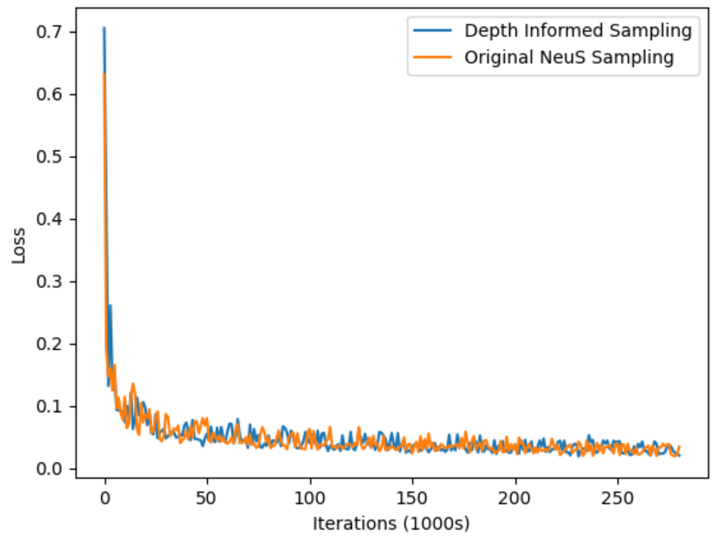


Figure 5.31: Loss over 300,000 iterations for the depth-informed and original NeuS sampling methods.

However, the 32 sample mesh acquired through our depth-informed sampling strategy (see Figure 5.33) appears to be of a higher quality than the 32 sample mesh acquired through the original NeuS method (see Figure 5.34). The mesh acquired through our method also appears of a similar quality to the mesh produced with 64 ray samples, shown in Figure 5.32, despite only using half as many ray samples.

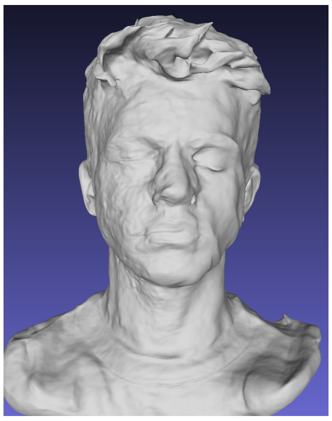


Figure 5.32: Surface from 64 samples and no depth-informed sampling.

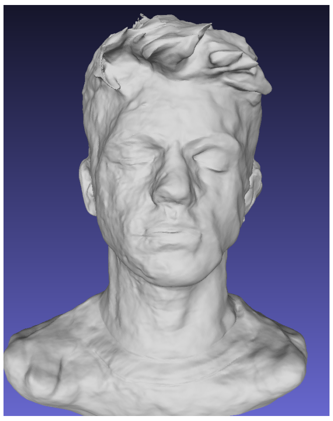


Figure 5.33: Surface from 32 samples and depth-informed sampling.

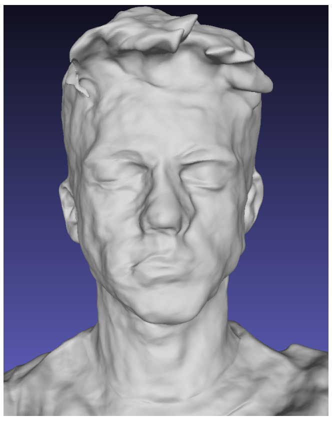


Figure 5.34: Surface from 32 samples and no depth-informed sampling.

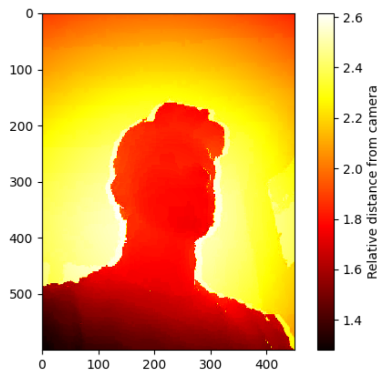


Figure 5.35: Heatmap of our depth informed midpoint estimation.

One flaw in the quality with the depth-informed meshes is that they are slightly narrower than those achieved with the original NeuS implementation, for example see the right temple in Figure 5.33. The likely reason for this is due to the sudden depth readings that occur around the edges of a subject in the depth maps. This means that when these depth maps are combined with the NeuS midpoint estimations, it results in a new midpoint estimation which is very far away (see the bright white pixels surrounding the head in Figure 5.35). Therefore, these areas around the edges of the subject are most likely being under-sampled. A solution, which future work could examine, would be to smooth the depth values around the edges of the subject, so that it corresponds with the depth of the rest of the background.

5.3 Comparison to Colmap

To evaluate the quality of the surface meshes acquired in this project, we will discuss how they compare to reconstructions achieved through using the Colmap software. Colmap, produced by Johannes Schoenberger in 2018 [60] uses structure from motion to produce a dense point cloud of an object, and then uses Poisson surface reconstruction [13] to connect these points into a polygon mesh. Colmap is used in our capture pipeline to extract camera poses from the set of input images. Hence, the majority of the capture and image processing pipeline is the same for Colmap as it is for our method which uses NeuS. However, instead of just using Colmap to extract camera poses, Colmap is instead used to create the final mesh (see Figure 5.36 for the Colmap pipeline alongside our pipeline).

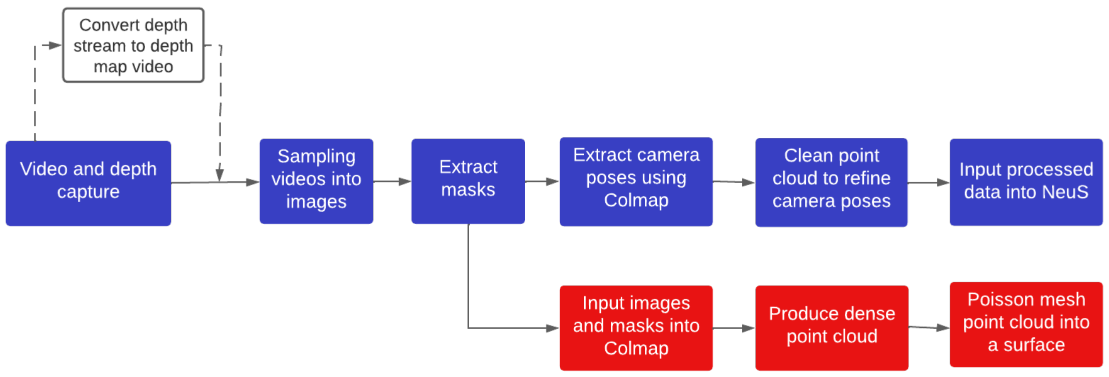

Convert depth stream to depth map video
Video and depth capture
Sampling videos into images
Extract masks
Extract camera poses using Colmap
Clean point cloud to refine camera poses
Input processed data into NeuS
Input images and masks into Colmap
Produce dense point cloud
Poisson mesh point cloud into a surface

Figure 5.36: NeuS image capture and processing pipeline in blue , next to Colmap capture and processing pipeline, shown in red .

The results of a facial geometry mesh acquired through Colmap reconstruction can be seen in Figure 5.37 and Figure 5.39. The mesh does a good job of rebuilding the core structure of the face, meaning key features such as the nose, ears and chin are distinguishable. However, there are clearly many errors in the surface creation. The Colmap reconstruction produces very noisy and uneven surfaces, particularly around areas with a higher density of interest points, such as the nose, mouth and eyebrows. Furthermore, the method fails to produce a complete mesh, as shown in Figure 5.37, the right cheek area has not been fully reconstructed, likely due to the lack of interest points detected in this area of plain skin. Another reason Colmap suffers to produce a complete mesh is due to the lack of input images. This reconstruction was built using only 23 images, whilst Colmap typically requires a much denser set of images to create accurate reconstructions. Hence, a clear downside of Colmap for reconstruction in these less controlled environments is its reliance on a large number of input images.

A NeuS reconstruction on the other hand, using our pipeline and the same input data produces the mesh shown in Figure 5.38 and Figure 5.40. Our reconstruction is able to produce a complete surface mesh, without any gaps. Additionally, whilst our surfaces do suffer from some inaccuracies and artifacts, they produce significantly smoother, more realistic surfaces than those produced via Colmap.

However, despite the lower quality surface produced by Colmap and structure from motion, one area in which Colmap does have an advantage is in processing speed. Producing a geometric mesh, such as the ones shown in Figures 5.37 and 5.39, takes between 25-40 minutes using Colmap with 23 images. This is significantly quicker than the 14-25 hour training times for NeuS.

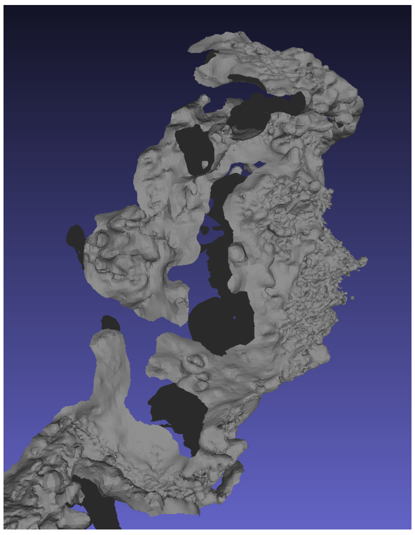


Figure 5.37: Side view of Poisson mesh acquired from Colmap.

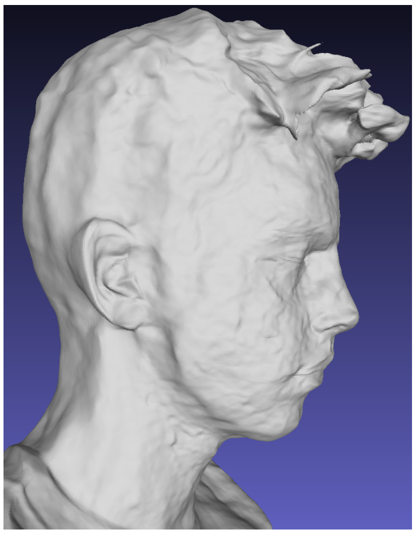


Figure 5.38: Side view of a mesh acquired from our NeuS pipeline.

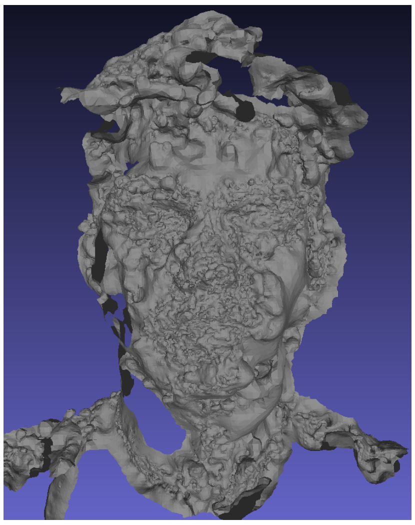


Figure 5.39: Frontal view of Poisson mesh acquired from Colmap.

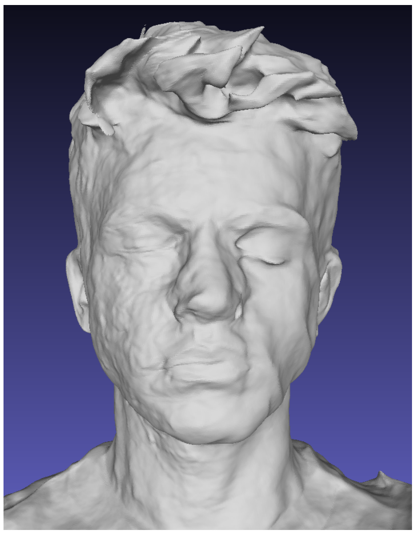


Figure 5.40: Frontal view of a mesh acquired from our NeuS pipeline.

Chapter 6

Conclusion

6.1 Conclusion

To conclude, this study aimed to experiment with using mobile devices to capture data for facial geometry rendering, primarily utilising the NeuS software for volumetric neural rendering. Several observations were made through the results of this study. Firstly, it appears that when using NeuS, training with masks has little effect of reconstruction accuracy. In fact, using masks can be harmful if the mask is not properly defined, resulting in the mesh fusing with the background. Instead, we found that a far more important factor was to correctly define the region of interest encapsulated by the camera poses. Additionally, as with state of the art reconstruction techniques, lighting conditions are still very influential in successful surface construction. More balanced, ambient, lighting conditions result in better surface reconstructions than lighting conditions that create heavy shadows and highly uneven pixel intensities. Despite these challenges in our pipeline, we still found that surface reconstructions achieved through this NeuS pipeline qualitatively outperformed surfaces acquired when using structure from motion with the same data.

Furthermore, one of the advantages offered by mobile devices is the access to depth sensing hardware. These depth cameras were used to inform our neural volumetric rendering sampling strategy, resulting in a quicker training time without reducing mesh quality. Although, even with our optimisation, one of the biggest limitations of this pipeline remains the long training times, which can range between 14 and 25 hours. However, with an increased availability of depth sensing capabilities on mobile devices, this work could be taken further to incorporate depth more to potentially be used to overcome some of the challenges faced in our pipeline.

6.2 Future Work

There are plenty of areas of further research which can be explored from this study. This project looked at a broad range of areas, including facial capture, image processing and neural volumetric rendering. Each of these topics could be explored in significantly more depth, and a few notable areas are listed below.

6.2.1 Incorporating Depth For Surface Accuracy

One of the most exciting areas for further study with this work is to build on the depth data incorporation. In this study, the depth data is used in mask creation and to reduce training times by informing the ray sampling strategy for volumetric rendering. However, further work could incorporate the depth data more into supervising the training of the MLP which encodes the SDF for the surface. Inspiration could be taken from other studies [54, 55, 56], who have also attempted to apply depth supervision into NeRF-based methods.

A greater utilisation of depth could also be used to potentially overcome some of the issues this study faced with its reliance on specific lighting conditions.

6.2.2 Incorporating Texture

Although the primary focus of this study has been on surface geometry reconstruction, a natural next step would be to incorporate texture onto the the 3D renders. This study could explore if the same challenges facing geometry reconstruction also impact texture acquisition.

6.2.3 Improving Image Quality

The capture method employed in this study used a video, taken by the user. This results in slightly blurred images. Therefore, further study could look to experiment with a higher image quality and see how this impacted results.

6.2.4 Instant NGP

Recent work by Müller et al [61], a group of researchers from NVIDIA, creates a method for near-instant training of neural graphics primitives using multi-resolution hash tables. This could be an exciting area of research to overcome the time limitations of our NeuS method.

6.2.5 Experimenting With Different Devices

This study used a large, 13 inch iPad Pro for capture. Whilst this device was suitable, similar results could be achieved on the latest generation of iPhone, as these phones also have depth sensing hardware. Using these smaller devices might make the capture process easier for the user, and reduce the micro-movement of the user whilst moving the device.

Bibliography

[1] Frederick Parke. A Parametric Model for Human Faces | Computer Science Archive, 1974. URL https://collections.lib.utah.edu/details?id=104762. pages 1, 11

[2] J. Eisenfeld, David R. Barker, and David J. Mishelevich. Iconic representation of the human face with computer graphics. *ACM SIGGRAPH Computer Graphics*, 8(3):9–15, September 1974. ISSN 0097-8930. doi: 10.1145/988026.988028. URL https://doi.org/10.1145/988026.988028. pages 1

[3] Bloomberg. Visualization & 3D Rendering Software Market Size is Projected To Reach USD 3083.9 Million By 2027, At A CAGR of 12.7% -. *Bloomberg.com*, January 2022. URL shorturl.at/iJT16. pages 1

[4] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics*, 37(4): 68:1–68:13, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201401. URL https://doi.org/10.1145/3197517.3201401. pages 1, 11, 12

[5] Abhijeet Ghosh, Tim Hawkins, Pieter Peers, Sune Frederiksen, and Paul Debevec. Practical Modeling and Acquisition of Layered Facial Reflectance. *ACM Transactions on Graphics (TOG)*, 27:139, December 2008. ISSN 978-1-4503-1831-0. doi: 10.1145/1409060.1409092. pages 1, 2, 12

[6] Stephen R. Marschner, Brian Guenter, and Sashi Raghupathy. Modeling and Rendering for Realistic Facial Animation. In Bernard Péroche and Holly Rushmeier, editors, *Rendering Techniques 2000*, pages 231–242. Springer Vienna, Vienna, 2000. ISBN 978-3-211-83535-7 978-3-7091-6303-0. doi: 10.1007/978-3-7091-6303-0_21. URL http://link.springer.com/10.1007/978-3-7091-6303-0_21. Series Title: Eurographics. pages 1, 11

[7] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 145–156. ACM Press, 2000. ISBN 978-1-58113-208-3. doi: 10.1145/344779.344855. URL http://portal.acm.org/citation.cfm?doid=344779.344855. pages 1, 2, 12

[8] Wan-Chun Ma, Tim Hawkins, Pieter Peers, Charles-Felix Chabert, Malte Weiss, and Paul Debevec. Rapid acquisition of specular and diffuse normal maps from

polarized spherical gradient illumination. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, EGSR'07, pages 183–194, Goslar, DEU, June 2007. Eurographics Association. ISBN 978-3-905673-52-4. pages 2, 12

[9] The Light Stages, . URL `https://vgl.ict.usc.edu/LightStages/`. pages 2

[10] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 405–421, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58452-8. doi: 10.1007/978-3-030-58452-8_24. pages 2, 13, 14

[11] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5569–5579, Montreal, QC, Canada, October 2021. IEEE. ISBN 978-1-66542-812-5. doi: 10.1109/ICCV48922.2021.00554. URL `https://ieeexplore.ieee.org/document/9709919/`. pages 3, 15

[12] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction, December 2021. URL `http://arxiv.org/abs/2106.10689`. arXiv:2106.10689 [cs]. pages 3, 22, 23

[13] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing*, pages 61–70, 2006. pages 5, 38

[14] Open3D – A Modern Library for 3D Data Processing, . URL `http://www.open3d.org/`. pages 6

[15] Soulaiman Hazzat, Abderrahim Saaidi, and Khalid Satori. Multi-view passive 3D reconstruction: Comparison and evaluation of three techniques and a new method for 3D object reconstruction. *International Conference on Next Generation Networks and Services, NGNS*, pages 194–201, December 2014. doi: 10.1109/NGNS.2014.6990252. pages 6

[16] Matthew Zucker, James Bagnell, Christopher Atkeson, and James Kuffner. An Optimization Approach to Rough Terrain Locomotion. pages 3589–3595, May 2010. doi: 10.1109/ROBOT.2010.5509176. pages 7

[17] Lars Meschedar, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. pages 7

[18] Volume ray casting, July 2022. URL `https://en.wikipedia.org/w/index.php?title=Volume_ray_casting&oldid=1097049152`. Page Version ID: 1097049152. pages 8

[19] Thomas Porter and Tom Duff. Compositing digital images. *ACM SIGGRAPH Computer Graphics*, 18(3):253–259, January 1984. ISSN 0097-8930. doi: 10.1145/964965.808606. URL `https://doi.org/10.1145/964965.808606`. pages 8

[20] Shweta Kadam. Neural Network Part1:Inside a single neuron, June 2020. URL `https://medium.com/analytics-vidhya/neural-network-part1-inside-a-single-neuron-fee5e44f1e`. pages 9

[21] Snehanshu Saha. *PERFORMANCE OF NOVEL ACTIVATION FUNCTIONS ON DEEP LEARNING ARCHITECTURE*. December 2021. pages 9

[22] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL `http://link.springer.com/10.1023/B:VISI.0000029664.99615.94`. pages 10

[23] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, Lecture Notes in Computer Science, pages 404–417, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-33833-8. doi: 10.1007/11744023_32. pages 10

[24] Kai Wu | Triangulation, . URL `https://imkaywu.github.io//blog/2017/07/triangulation/`. pages 10

[25] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, pages 187–194. ACM Press, 1999. ISBN 978-0-201-48560-8. doi: 10.1145/311535.311556. URL `http://portal.acm.org/citation.cfm?doid=311535.311556`. pages 11, 13

[26] Yuencheng Lee, Demetri Terzopoulos, and Keith Walters. Realistic modeling for facial animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95*, pages 55–62. ACM Press, 1995. ISBN 978-0-89791-701-8. doi: 10.1145/218380.218407. URL `http://portal.acm.org/citation.cfm?doid=218380.218407`. pages 11

[27] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Fredric Pighin. Making faces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 55–66, New York, NY, USA, July 1998. Association for Computing Machinery. ISBN 978-0-89791-999-9. doi: 10.1145/280814.280822. URL `https://doi.org/10.1145/280814.280822`. pages 11, 12

[28] P. Fua and C. Miccio. From regular images to animated heads: A least squares approach. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Hans Burkhardt, and Bernd Neumann, editors, *Computer Vision — ECCV'98*, volume 1406, pages 188–202. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-64569-6 978-3-540-69354-3. doi: 10.1007/BFb0055667. URL `http://link.springer.com/10.1007/BFb0055667`. Series Title: Lecture Notes in Computer Science. pages 11

[29] Yu Yanga, Xiao-Jun Wu, and Josef Kittler. Landmark Weighting for 3DMM Shape Fitting. Technical Report arXiv:1808.05399, arXiv, August 2018. URL `http://arxiv.org/abs/1808.05399`. arXiv:1808.05399 [cs] type: article. pages 11

[30] Victoria Fernández Abrevaya, Stefanie Wuhrer, and Edmond Boyer. Multilinear Autoencoder for 3D Face Model Learning. In *WACV 2018 - IEEE Winter Conference on Applications of Computer Vision*, pages 1–9, Lake Tahoe, NV/CA, United States, March 2018. IEEE. doi: 10.1109/WACV.2018.00007. URL `https://hal.archives-ouvertes.fr/hal-01700934`. pages 11, 12

[31] Alexandros Lattas, Stylianos Moschoglou, Baris Gecer, Stylianos Ploumpis, Vasileios Triantafyllou, Abhijeet Ghosh, and Stefanos Zafeiriou. AvatarMe: Realistically Renderable 3D Facial Reconstruction ”in-the-wild”, March 2020. URL `http://arxiv.org/abs/2003.13845`. arXiv:2003.13845 [cs]. pages 11, 12, 13

[32] Bernhard Egger, William A. P. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter. 3D Morphable Face Models – Past, Present and Future, April 2020. URL `http://arxiv.org/abs/1909.01815`. arXiv:1909.01815 [cs]. pages 12

[33] Gil Shamai, Ron Slossberg, and Ron Kimmel. Synthesizing facial photometries and corresponding geometries using generative adversarial networks, January 2019. URL `http://arxiv.org/abs/1901.06551`. arXiv:1901.06551 [cs]. pages 12, 13

[34] Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. High Resolution Passive Facial Performance Capture. *ACM Trans. Graph.*, 29, July 2010. doi: 10.1145/1833349.1778778. pages 12

[35] M. Shahriar Hossain, Monika Akbar, and J. Denbigh Starkey. Inexpensive construction of a 3D face model from stereo images. In *2007 10th international conference on computer and information technology*, pages 1–6, December 2007. doi: 10.1109/ICCITECHN.2007.4579387. pages 12

[36] Christos Kampouris, Stefanos Zafeiriou, and Abhijeet Ghosh. Diffuse-specular separation using binary spherical gradient illumination. In *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations*,

SR '18, pages 1–10, Goslar, DEU, July 2018. Eurographics Association. doi: 10.2312/sre.20181167. URL `https://doi.org/10.2312/sre.20181167`. pages 12

[37] Jérémy Riviere, Paulo Gotardo, Derek Bradley, Abhijeet Ghosh, and Thabo Beeler. Single-shot high-quality facial geometry and skin appearance capture. *ACM Transactions on Graphics*, 39(4), August 2020. ISSN 0730-0301, 1557-7368. doi: 10.1145/3386569.3392464. URL `https://dl.acm.org/doi/10.1145/3386569.3392464`. pages 12

[38] Abhijeet Ghosh, Graham Fyffe, Borom Tunwattanapong, Jay Busch, Xueming Yu, and Paul Debevec. Multiview face capture using polarized spherical gradient illumination. *ACM Transactions on Graphics*, 30(6):1–10, December 2011. ISSN 0730-0301. doi: 10.1145/2070781.2024163. URL `https://doi.org/10.1145/2070781.2024163`. pages 12

[39] Steffen Herbort and Christian Wöhler. An introduction to image-based 3D surface reconstruction and a survey of photometric stereo methods. *3D Res*, 2:1–17, September 2011. doi: 10.1007/3DRes.03(2011)4. pages 12

[40] Thabo Beeler, Bernd Bickel, Gioacchino Noris, Paul Beardsley, Steve Marschner, Robert W. Sumner, and Markus Gross. Coupled 3D reconstruction of sparse facial hair and skin. *ACM Transactions on Graphics*, 31(4):117:1–117:10, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185613. URL `https://doi.org/10.1145/2185520.2185613`. pages 12

[41] Levi Valgaerts, Chenglei Wu, Andrés Bruhn, Hans-Peter Seidel, and Christian Theobalt. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Transactions on Graphics*, 31(6):187:1–187:11, November 2012. ISSN 0730-0301. doi: 10.1145/2366145.2366206. URL `https://doi.org/10.1145/2366145.2366206`. pages 12

[42] G. Fyffe, P. Graham, B. Tunwattanapong, A. Ghosh, and P. Debevec. Near-instant capture of high-resolution facial geometry and reflectance. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics*, EG '16, pages 353–363, Goslar, DEU, May 2016. Eurographics Association. pages 12

[43] Eugene d'Eon, David Luebke, and Eric Enderton. Efficient rendering of human skin. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, EGSR'07, pages 147–157, Goslar, DEU, June 2007. Eurographics Association. ISBN 978-3-905673-52-4. pages 12

[44] Prashanth Chandran, Sebastian Winberg, Gaspard Zoss, Jérémy Riviere, Markus Gross, Paulo Gotardo, and Derek Bradley. Rendering with style: combining traditional and neural approaches for high-quality face rendering. *ACM Transactions on Graphics*, 40(6):1–14, December 2021. ISSN 0730-0301, 1557-7368. doi: 10.1145/3478513.3480509. URL `https://dl.acm.org/doi/10.1145/3478513.3480509`. pages 13

[45] Eduard Ramon, Janna Escur, and Xavier Giro-i Nieto. Multi-View 3D Face Reconstruction in the Wild Using Siamese Networks. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3096–3100, Seoul, Korea (South), October 2019. IEEE. ISBN 978-1-72815-023-9. doi: 10.1109/ICCVW.2019.00373. URL `https://ieeexplore.ieee.org/document/9022189/`. pages 13

[46] Chen Cao, Vasu Agrawal, Fernando De La Torre, Lele Chen, Jason Saragih, Tomas Simon, and Yaser Sheikh. Real-time 3D neural facial animation from binocular video. *ACM Transactions on Graphics*, 40(4):1–17, August 2021. ISSN 0730-0301, 1557-7368. doi: 10.1145/3450626.3459806. URL `https://dl.acm.org/doi/10.1145/3450626.3459806`. pages 13

[47] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN, March 2020. URL `http://arxiv.org/abs/1912.04958`. arXiv:1912.04958 [cs, eess, stat]. pages 13

[48] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction, December 2020. URL `http://arxiv.org/abs/2012.03065`. arXiv:2012.03065 [cs]. pages 13

[49] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. FENeRF: Face Editing in Neural Radiance Fields, March 2022. URL `http://arxiv.org/abs/2111.15490`. arXiv:2111.15490 [cs]. pages 13

[50] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image Synthesis, October 2021. URL `http://arxiv.org/abs/2110.08985`. arXiv:2110.08985 [cs, stat]. pages 13

[51] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in Neural Rendering, March 2022. URL `http://arxiv.org/abs/2111.05849`. arXiv:2111.05849 [cs]. pages 13

[52] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields, August 2021. URL `http://arxiv.org/abs/2103.13415`. arXiv:2103.13415 [cs]. pages 15

[53] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs, August 2021. URL `http://arxiv.org/abs/2103.13744`. arXiv:2103.13744 [cs]. pages 15

[54] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense Depth Priors for Neural Radiance Fields from Sparse Input Views, April 2022. URL `http://arxiv.org/abs/2112.03288`. arXiv:2112.03288 [cs]. pages 15, 41

[55] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free, April 2022. URL `http://arxiv.org/abs/2107.02791`. arXiv:2107.02791 [cs]. pages 15, 41

[56] David Dadon, Ohad Fried, and Yacov Hel-Or. DDNeRF: Depth Distribution Neural Radiance Fields, March 2022. URL `http://arxiv.org/abs/2203.16626`. arXiv:2203.16626 [cs]. pages 15, 41

[57] Cheng-Wei Wang and Chao-Chung Peng. 3D Face Point Cloud Reconstruction and Recognition Using Depth Sensor. *Sensors*, 21(8):2587, January 2021. ISSN 1424-8220. doi: 10.3390/s21082587. URL `https://www.mdpi.com/1424-8220/21/8/2587`. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute. pages 15

[58] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance, October 2020. URL `http://arxiv.org/abs/2003.09852`. arXiv:2003.09852 [cs]. pages 15

[59] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5202–5211, June 2020. doi: 10.1109/CVPR42600.2020.00525. ISSN: 2575-7075. pages 20, 27

[60] Johannes L. Schönberger. *Robust Methods for Accurate and Efficient 3D Modeling from Unstructured Imagery*. Doctoral Thesis, ETH Zurich, 2018. URL `https://www.research-collection.ethz.ch/handle/20.500.11850/295763`. Accepted: 2018-10-15T06:22:19Z. pages 22, 38

[61] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530127. URL `http://arxiv.org/abs/2201.05989`. arXiv:2201.05989 [cs]. pages 41